



## What's New in 12d Model 11

**Version 11**

**August 2015**

**12D SOLUTIONS PTY LTD**

ACN 101 351 991

PO Box 351 Narrabeen NSW Australia 2101

Australia Telephone (02) 9970 7117 Fax (02) 9970 7118

International Telephone 61 2 9970 7117 Fax 61 2 9970 7118

email [support@12d.com](mailto:support@12d.com) web [www.12d.com](http://www.12d.com)



---

# What's New in 12d Model 11

## Disclaimer

12d Model is supplied without any express or implied warranties whatsoever.

No warranty of fitness for a particular purpose is offered.

No liabilities in respect of engineering details and quantities produced by 12d Model are accepted.

Every effort has been taken to ensure that the advice given in this manual and the program 12d Model is correct, however, no warranty is expressed or implied by 12d Solutions Pty Ltd.

## Copyright

This manual is copyrighted and all rights reserved.

This manual may not, in whole or part, be copied or reproduced without the prior consent in writing from 12d Solutions Pty Ltd.

Copies of 12d Model software must not be released to any party, or used for bureau applications without the written permission of 12D Solutions Pty Ltd.

Copyright (c) 1989-2015 by 12d Solutions Pty Ltd

Sydney, New South Wales, Australia.

ACN 101 351 991

All rights reserved.



# Table of Contents

<b>1. Moving to 12d Model 11</b> .....	<b>15</b>
1.1 About This Manual .....	15
1.2 Some Important Notes on 12d Model 11 .....	16
1.3 Dongles for 12d Model 11 .....	17
1.4 New Installers and Uninstallers .....	19
1.4.1 Installers for Wibu and CodeMeter Drivers .....	19
1.4.2 Installers for 12d Model.....	20
1.4.3 Uninstallers for 12d Model.....	22
1.5 User, User_Lib and env.4d for 12d Model 11 .....	23
1.5.1 User.....	23
1.5.2 User_Lib .....	24
1.5.3 env.4d.....	25
1.5.4 nodes.4d.....	25
1.6 Nodes.4d is XML .....	26
1.7 Network Dongles and 12d Model 11 .....	27
<b>2. New Front Panel</b> .....	<b>28</b>
2.1 New Front Panel.....	28
2.2 Browse Button and Open Tab .....	29
2.3 New Button and New Tab.....	30
<b>3 New Objects</b> .....	<b>31</b>
3.1 Trimeshes .....	32
3.2 Dimensions.....	34
3.3 Leaders .....	35
3.4 Tables .....	36
<b>4. Special File Types</b> .....	<b>37</b>
4.1 12daz - Zipped .12da File.....	37
4.2 Label Map File .....	38
4.2.1 Active Column in Grids.....	38
4.3 Map File.....	39
4.3.1 Active Column in Grids.....	39
4.3.2 Attributes Node.....	40
4.3.3 Group Column in Grids .....	42
4.4 Names.4d File .....	43
4.4.1 Group Column in Names.4d .....	44
4.4.2 Active Column in Grids in Names.4d.....	46
4.4.3 Pipe Node in Names.4d .....	47
4.4.4 Attributes Node in Names.4d.....	48
<b>5. Drag and Drop</b> .....	<b>49</b>
5.1 File Box .....	49
5.2 .12daz, 12da Files.....	49
5.3 .dat Files .....	50
5.4 .chain,.rcn Files .....	50
5.5 .mtf Files .....	50
5.6 .mapfile Files.....	50
5.7 .slx Files .....	51
5.8 .12dpsf Files .....	51
5.9 .4do Files.....	51
5.10 .project Files.....	51
5.11 .dwg,.dxf Files.....	52
<b>6. Emailing from File Boxes</b> .....	<b>53</b>
<b>7. Setting Up XML Reports</b> .....	<b>55</b>
7.1 Create/Edit Report Types.....	56
7.1.1 [Create] for Report Types .....	58
7.1.2 [Edit] for Report Types.....	59
7.1.2.1 Creating New Report Types.....	59
7.1.2.2 Copy and Paste.....	61

7.1.2.3	Deleting a Report Type.....	62
7.1.2.4	Duplicating a Report Type.....	62
7.1.2.5	Editing a Report Type.....	63
<b>8.</b>	<b>Sharing .....</b>	<b>64</b>
8.1	Local Caching of Share Data .....	64
8.2	Share Manager .....	65
8.2.1	Making Models or Tins Available for Sharing .....	66
8.2.2	Sharing in Models and Tins.....	68
8.3	Sharing Map Files .....	71
8.4	Localize Shares .....	73
<b>9.</b>	<b>Smart Chainages .....</b>	<b>74</b>
9.1	Extension Value .....	74
9.2	Alias in Grids .....	75
9.3	Named Parts Accessing Vertical Parts.....	75
9.4	New Smart Chainage Modes .....	75
9.4.1	Intersection of 2 strings .....	77
9.4.2	Intersection of 2 strings - Replaced by Simple Version .....	78
9.4.3	Relative to Self End.....	79
9.4.4	Relative to Self Start.....	79
9.4.5	Relative to Alias Start.....	79
9.4.6	Relative to Alias End.....	79
9.4.7	Relative to Previous Start .....	80
9.4.8	Relative to Previous End .....	80
9.4.9	Drop Point to Reference String .....	81
9.4.10	Drop Point to Other String .....	81
9.4.11	Lowest Dropped Chainage of Modifier String.....	82
9.4.12	Highest Dropped Chainage of Modifier String .....	82
9.4.13	Snippet Cut Ref with Model of Strings .....	82
9.4.14	Snippet Placed to Model of Strings .....	82
<b>10.</b>	<b>Chains.....</b>	<b>83</b>
10.1	Some Options That Didn't Record & Now Can .....	84
10.2	Warning Prompt Tick Box.....	84
10.3	Command Name Moved .....	85
10.4	Icons for Activate and Deactivate.....	85
10.5	Clicking on a Resolve Command Highlights the SA .....	86
10.6	You Get a Warning When Leaving a Project and a Chain is Still Open .....	86
10.7	Clicking Command Name Icon Regenerates Name .....	86
10.8	Re-Recording Does Not Change the Command Name .....	86
10.9	There is a Preview Button for Functions .....	86
10.10	Recording an Apply MTF.....	86
10.11	Disable Delete Confirmation .....	86
10.12	Copy and Paste Between Chains .....	87
10.13	Chain View Commands .....	87
10.14	Creating a View Giving Top Left and Bottom Right .....	87
10.15	Dump Image of a View.....	88
10.16	Delete Function Command .....	89
10.17	Delete All Functions Command .....	89
10.18	Prompt Command .....	89
10.18.1	Prompt Title.....	89
10.18.2	Prompt Coming Up at the Same Place .....	89
10.19	Output Window Commands .....	90
10.20	Sleep Command.....	90
10.21	Save Project .....	90
10.22	Copy Command .....	90
10.23	Dump Image of SLX of a Panel.....	92
10.24	Shell Command Now has Separate Argument Line .....	94
10.25	Command - If Function Exists.....	94

10.26	Command - If Parameter Equals - Parameter Pop Up .....	94
10.27	Commands for Multiple Lines .....	95
10.28	Chain Parameters Editor .....	96
10.28.1	Icons for Parameter Types .....	96
10.28.2	Comments .....	96
10.28.3	Search.....	96
10.28.4	Set No Longer Needed.....	96
10.29	Chain Parameters as a 12dPL Arguments.....	97
10.30	Printing the Value of a Chain Parameter.....	98
10.31	Append for PPF Grid Parameters.....	99
10.32	PVF Substitution in User Menus and Toolbars.....	101
<b>11</b>	<b>BIM .....</b>	<b>102</b>
11.1	What is BIM ? .....	102
11.2	The 12d Approach to BIM .....	104
11.3	The UK Government and BIM .....	107
11.4	Common Terms in BIM .....	108
11.5	What is Wrong with Local Coordinates ? .....	111
11.5.1	An Exact Position for Everything on the Earth .....	112
11.5.2	Map (Cartographic) Projections - Eastings and Northings.....	113
11.5.3	Why Isn't MGA Used on a Small Building Site ?.....	114
11.6	IFCs.....	115
11.6.1	Spatial Structures .....	116
11.6.2	IFC Definitions .....	118
11.6.2.1	ifcProject .....	119
11.6.2.2	ifcSite .....	119
11.6.2.3	ifcElement .....	120
11.6.2.4	ifcSpatialStructureElement .....	120
11.6.2.5	ifcBuildingElement .....	121
11.6.2.6	ifcBuildingElementProxy.....	121
11.6.2.7	ifcFlowStorageDevice.....	121
11.6.2.8	ifcFlowStorageDeviceType.....	121
11.6.2.9	ifcFlowSegment .....	121
11.6.2.10	ifcFlowSegmentType .....	122
11.6.2.11	ifcRelContainedInSpatialStructure .....	123
11.6.2.12	ifcRelAggregates.....	123
11.6.2.13	ifcProduct.....	124
11.6.2.14	ifcPropertySet.....	124
11.6.2.15	IfcPropertySingleValue.....	124
11.6.2.16	IfcShapeRepresentation.....	125
11.6.2.17	IfcCircleHollowProfileDef.....	125
11.6.2.18	IfcRectangleHollowProfileDef .....	126
11.6.3	Representation of 12d Objects as IFCs.....	128
11.6.3.1	Super Alignment .....	128
11.6.3.2	Alignment.....	128
11.6.3.3	Drainage String .....	128
11.6.3.4	Sewer String.....	129
11.6.3.5	Tin .....	129
11.6.3.6	Trimesh .....	129
<b>12</b>	<b>Bits and Pieces.....</b>	<b>130</b>
12.1	Search Looks at Full Menu Path .....	131
12.2	Moving the Right Hand Side in Tree Controls .....	132
12.3	Names.4d Comments in Data Tooltips .....	133
12.4	New Plotter Type for PDF Plots .....	134
12.5	Additions to Plotters.4d.....	135
12.5.1	pdf_12d Plotter Engine .....	135
12.5.2	string_weights Keyword.....	135
12.5.3	Group Keyword .....	136

12.6	String/Model Name & Id Used in DXF, SLX Files .....	137
12.7	String/Model Name & Id Used in Section View .....	137
12.8	Demand Loading of Billboards & Textures .....	137
12.9	MB in View Panel Field to Get View .....	138
12.10	Resizing Panels .....	138
12.11	Regions in Some Grids .....	139
12.12	Extra Options on Model Icon Pop Up .....	140
12.13	Extra Options on Tin Icon Pop Up .....	141
12.14	Pipe ControlBar .....	142
12.15	Attributes ControlBar .....	143
12.16	Icons in Property Sheets .....	143
12.17	More Borders Types for Text .....	144
12.18	Arithmetic in an Integer Panel Field.....	144
12.19	If Else in Real Value Fields .....	145
12.20	Legacy Name in Colours.4d .....	146
12.21	Colours in Pop Up Order .....	147
12.21.1	SORT_COLOURS_BY_POPUP_NUM_4D.....	147
12.21.2	Setting SORT_COLOURS_BY_POPUP_NUM_4D.....	149
12.22	Full Stop Allowed in Colour Name .....	149
12.23	Subgroups in Linestyles.4d and Symbols.4d.....	150
<b>13.</b>	<b>Menus on Views.....</b>	<b>151</b>
13.1	Text for Z Values, Point Ids etc on Plan Views.....	152
13.1.1	Text for Z Values etc on Plan Views Demystified.....	152
13.1.1.1	Single.....	153
13.1.1.2	Table.....	153
13.1.1.3	Toggle Walk-Right.....	155
13.1.1.4	Toggle Menu .....	156
13.1.2	Text for Z Values etc on Plan Views - Single .....	157
13.1.3	Text for Point Id etc on Plan Views - Table .....	159
13.2	Section View 2 Points Profile .....	162
13.3	(x,y) in Section View .....	162
13.4	View Properties Panel.....	164
13.5	Clone a View .....	165
<b>14.</b>	<b>Project Menu .....</b>	<b>166</b>
14.1	Projects =>Utilities =>Zip Removed.....	167
14.2	New Projects =>Recent projects.....	167
14.3	Modified Projects =>Open.....	167
14.4	Modified Projects =>New.....	167
14.5	Management.....	168
14.5.1	Modified Defaults.....	169
14.5.2	Modified env.4d .....	170
14.5.2.1	MTF and Boxing.....	170
14.5.3	Dongles.....	171
14.5.3.1	12d Dongles, Dongles.4d, Nodes.4d, and Authorizing .....	172
14.5.3.2	Dongles Administration.....	174
14.5.3.3	Dongles.4d Editor .....	178
14.5.3.4	Certify CodeMeter .....	183
14.5.3.5	Authorization Details .....	184
14.5.3.6	Nodes.4d Editor .....	188
14.5.3.7	CodeMeter Network User.....	193
14.5.4	Managers .....	194
14.5.4.1	Project Manager.....	195
14.5.4.2	Model Manager.....	197
14.5.4.3	Tin Manager.....	198
14.5.4.4	Function Manager .....	199
14.5.4.5	Template Manager .....	200
14.5.4.6	View Manager.....	201

14.5.5	Toggle Topmost Buttons .....	202
<b>15.</b>	<b>File Menu.....</b>	<b>203</b>
15.1	Data Input.....	204
15.1.1	12d Archive Input .....	205
15.1.2	Read 12d XML Data.....	208
15.1.3	12d XML to 12da files.....	210
15.1.4	12d XML to 12da file .....	211
15.1.5	DWG PolyFace Mesh to Trimesh.....	212
15.1.6	Import 2D PDF .....	213
15.1.7	Wavefront OBJ Input.....	219
15.1.8	Input Point Cloud Files .....	220
15.1.8.1	E57 Cloud Input .....	221
15.1.8.2	Faro Point Cloud Input.....	228
15.1.8.3	LAS Cloud Input .....	237
15.1.8.4	Leica Point Cloud Input .....	256
15.1.8.5	Read SRTM Files .....	273
15.2	Data Output .....	277
15.2.1	12d Archive Output .....	278
15.2.2	12d XML Data .....	281
15.2.3	12d XML Project Data.....	284
15.2.4	Trimesh to DWG .....	286
15.2.5	Write 3D PDF .....	287
15.2.5.1	How to View the Generated 3D PDF.....	288
15.2.6	Export DAE .....	289
15.2.7	IFC Output .....	290
15.2.7.1	IFC 2x3 Express Writer .....	291
15.2.7.2	Project Units.....	293
15.2.7.3	Extra Project Details .....	295
15.2.7.4	Defining and Assigning Spatial Structures .....	299
15.2.7.5	IFC Writer .....	305
15.2.8	Export OBJ .....	307
15.2.9	Export STL .....	309
15.3	Map Files =>Apply .....	310
15.4	Range File Menu .....	311
15.4.1	Area Range File .....	311
<b>16.</b>	<b>Models Menu.....</b>	<b>313</b>
16.1	Copy Folder Models.....	314
<b>17.</b>	<b>Strings Menu .....</b>	<b>317</b>
17.1	Texts Edits.....	317
17.1.1	Change Text Colour.....	318
17.1.2	Change Text Height.....	318
17.1.3	Change Text Style.....	318
17.1.4	Change Text Angle .....	319
17.1.5	Change Text Justify .....	319
17.1.6	Change Combine.....	320
17.2	Multi String Properties.....	322
17.3	Strings =>Create =>Super.....	324
17.4	Reverse Only Geometry of SA .....	330
17.5	Offset Mode in Parallel SA.....	331
17.6	Trimesh .....	332
17.6.1	Trimeshes from 12d Objects.....	333
17.6.2	Trimeshes from Wavefront OBJ File.....	335
17.6.3	Trimeshes from FBX File.....	336
17.6.4	Trimeshes from Tin .....	337
17.6.5	Trimeshes from Polygon.....	338
17.6.6	Trimesh Tools.....	340
17.6.6.1	Trimesh Flip Faces.....	340

17.6.7	Generate Trimesh from 12d Objects .....	341
17.6.8	Generate Trimesh from Tin .....	343
17.7	Super String Fills .....	344
17.8	Strings =>Label =>Cut/fill .....	345
<b>18. Cad Menu.....</b>	<b>.....</b>	<b>346</b>
18.1	CAD Dimension .....	347
18.1.1	Dimension Association.....	349
18.1.2	Continuous Mode .....	350
18.1.3	Baseline Mode.....	352
18.1.4	Zero Offset Mode .....	354
18.1.5	Dimension - Create Same As .....	355
18.1.6	Linear Dimension Commands.....	356
18.1.6.1	Linear .....	358
18.1.6.2	Aligned.....	360
18.1.6.3	Horizontal .....	361
18.1.6.4	Vertical.....	363
18.1.6.5	Rotated.....	365
18.1.6.6	Aligned Segment.....	367
18.1.6.7	Horizontal Segment .....	368
18.1.6.8	Vertical Segment.....	370
18.1.6.9	Rotated Segment .....	372
18.1.7	Length.....	374
18.1.8	Drop Segment.....	378
18.1.9	Drop String .....	381
18.1.10	Area .....	382
18.1.11	Jogged radius .....	383
18.1.12	Diameter .....	384
18.1.13	Radius .....	386
18.1.14	Angle 2 Lines .....	387
18.1.15	Angle 3 points.....	390
18.1.16	Angle Arc .....	392
18.1.17	Arc Length by Centre, 2 Points .....	393
18.1.18	Dimension Utilities.....	395
18.1.18.1	Dimension Edit .....	396
18.1.18.2	Dimension Styles .....	398
18.1.18.3	Change Style of Dimension .....	406
18.1.18.4	Change Text Format .....	407
18.1.18.5	Create Many Segment Lengths.....	408
18.2	CAD Leader .....	410
18.2.1	Leader Association .....	412
18.2.2	Hook Angle .....	413
18.2.3	Leader - Create Same As.....	414
18.2.4	Text Leader.....	415
18.2.5	Information Leader .....	417
18.2.5.1	Label .....	419
18.2.5.2	Area.....	420
18.2.5.3	String Length .....	422
18.2.5.4	String Length 3D .....	423
18.2.5.5	Segment Length.....	424
18.2.5.6	Segment Length 3D .....	425
18.2.5.7	Segment Bearing.....	426
18.2.5.8	Bearing & Segment Length .....	427
18.2.5.9	Segment Radius .....	428
18.2.5.10	String name .....	430
18.2.5.11	Vertex Text .....	431
18.2.5.12	Segment Text.....	432
18.2.5.13	Vertex XYZ .....	433
18.2.5.14	Vertex XY.....	434

18.2.5.15	String XYZ.....	435
18.2.5.16	String XY.....	436
18.2.5.17	String Z.....	437
18.2.5.18	Tin Z.....	438
18.2.5.19	Tin Depth.....	440
18.2.5.20	Grade (%).....	442
18.2.5.21	Grade 1 n (Slope).....	443
18.2.5.22	Centroid XY.....	444
18.2.5.23	Medial Centre XY.....	446
18.2.5.24	Trimesh Volume.....	448
18.2.5.25	Trimesh Surface Area.....	449
18.2.6	Leader Utilities.....	450
18.2.6.1	Leader Edit.....	451
18.2.6.2	Leader Styles.....	453
18.2.6.3	Change Style of Leader.....	460
18.3	Text Format for Dimensions and Leaders.....	461
18.3.1	Angle Brackets <>.....	462
18.3.2	Pre and Post Text.....	463
18.3.3	Units Factor.....	463
18.3.4	Round Off.....	464
18.3.5	Number of Decimals.....	464
18.3.6	Dynamic Text Format.....	465
18.3.7	Multiple Text Formats.....	465
18.3.8	Typing t or T to Change the Text Format.....	466
18.3.9	Defaults File for Text Formats.....	467
18.3.9.1	An Excerpt from a draft_text.xml File.....	468
18.4	CAD Table.....	469
18.4.1	Create Table.....	470
18.4.2	Create Table from a Super Alignment.....	472
18.4.3	Create Table from a CSV File.....	475
18.4.4	Table Edit.....	477
18.4.4.1	Move.....	478
18.4.4.2	Column and Row Edits.....	479
18.4.4.3	Text and Values Edit.....	484
18.4.4.4	Fit.....	490
18.4.4.5	Recalc.....	493
18.4.4.6	Properties and Associations.....	494
18.4.4.7	Content.....	498
18.4.5	Table Styles.....	499
18.4.5.1	Table Styles - General Tab.....	501
18.4.5.2	Table Styles - Title Style Node.....	501
18.4.5.3	Table Styles - Header Style Node.....	503
18.4.5.4	Table Styles - Data Style Node.....	505
18.4.6	Table to CSV.....	506
18.5	Style XML Files for Dimensions, Leaders and Tables.....	507
18.5.1	Create & Editing Dimensions, Leader and Table Styles.....	508
18.5.1.1	[Create] for Dimensions, Leader and Table Styles.....	509
18.5.1.2	[Edit] for Dimensions, Leader and Table Styles.....	509
<b>19.</b>	<b>Tins Menu.....</b>	<b>516</b>
19.1	127 Tins in a Super Tin.....	516
19.2	Exact Calculations Flag for a Super Tin.....	516
19.3	Colour by Triangle Data.....	516
19.4	New Tin Manager.....	517
19.5	New Recalc Super Tin.....	517
19.6	New Depth, Contour and Label Function.....	519
<b>20</b>	<b>Drainage, Sewer and Tuflow.....</b>	<b>522</b>
<b>21</b>	<b>Volumes.....</b>	<b>523</b>

21.1	Trimesh .....	524
21.1.1	Trimesh Volume and Area Report .....	525
21.1.2	Trimesh Volume Along a String .....	527
<b>22.</b>	<b>Design Menu .....</b>	<b>529</b>
22.1	Templates .....	530
22.1.1	Decisions on Template Now a Grid .....	530
22.1.2	Active Column in the Decisions Grid .....	531
22.1.3	Regions in the Decisions Grid .....	531
22.2	Apply Menu .....	532
22.2.1	Apply MTF .....	532
22.2.1.1	Smart Chainages in Apply MTF .....	533
22.2.1.2	Summary Volumes Written to Output Window .....	533
22.3	MTF .....	534
22.3.1	Snippets .....	534
22.3.1.1	Edit Snippet .....	535
22.3.1.2	Create Trimesh Snippet .....	541
22.3.1.3	Edit Snippet File .....	541
22.3.1.4	Compile Snippet .....	542
22.3.1.5	12d Supplied Snippets .....	543
22.4	Boxing .....	580
22.4.1	Create/Edit Definitions .....	580
22.4.1.1	Boxing Rules on Edit Boxing Definitions Now a Grid .....	580
22.4.1.2	Active Tick Box Column .....	581
22.4.1.3	Regions in the Boxing Rules Grid .....	581
22.4.2	Rename Boxing File .....	581
22.4.3	Copy Boxing .....	582
22.4.4	Delete Boxing .....	583
22.4.5	Smart Chainages in Boxing Many Function .....	584
22.4.5.1	Smart Chainages on Boxing Many Front Panel .....	584
22.4.5.2	Smart Chainages on Left/Right Boxing Tabs .....	584
22.5	Roads .....	586
22.5.1	Components .....	586
22.5.1.1	RIGHT TURN MEDIAN .....	586
22.5.1.2	MAJOR MINOR ROAD INTERSECTION .....	586
22.5.1.3	HAMMERHEAD .....	587
22.5.2	Create Roads - Enhanced .....	595
22.6	Sight Lines .....	604
22.6.1	Viewshed All .....	605
22.6.2	Shadow Analysis .....	609
22.6.3	Sight Distance Enhanced .....	613
22.7	Tunnels and Structures .....	618
22.7.1	Creating a Tunnel Definition File .....	619
22.7.2	Create Tunnel .....	625
22.7.3	Conform Tunnel .....	627
22.7.4	Conform Plot Tunnel .....	628
22.7.5	Plot Setting .....	630
22.7.5.1	Tunnel Plot Settings .....	631
22.7.5.2	Tunnel Plot Name Mappings .....	632
22.7.5.3	Tunnel Table Styles .....	633
22.7.6	Trimesh from Tunnel Points .....	634
22.8	Check/Clash .....	635
22.8.1	Clash Detection .....	636
22.8.2	Clash Detection Rules .....	638
<b>23.</b>	<b>MTF Edit .....</b>	<b>645</b>
23.1	Migrating MTFs from V10 to V11 .....	647
23.1.1	New Look for MTF Edit and Left/Right Modifiers .....	647
23.1.2	New MTF Modify Left/Right Link Commands .....	648

23.1.3	Automatic Conversion of V10 MTFs to V11 MTFs .....	649
23.2	Smart Chainages in MTF Edit.....	650
23.3	MTF Edit - Reorganised.....	651
23.4	MTF Edit =>Hinge.....	652
23.4.1	Tick Boxes in the Grid.....	653
23.4.2	Turning Off Extra Start/End .....	653
23.4.3	Env Variable to Turn Off Interval Column .....	653
23.4.4	Active Column in Hinge Grid.....	653
23.4.5	New Look for Hinge Modifier Panels .....	653
23.4.6	Autopan Button.....	654
23.4.7	Regions .....	654
23.4.8	Height To Tin.....	654
23.4.9	Smart Chainages for Hinge Modifiers .....	656
23.5	MTF Edit =>Modify Left/Right.....	657
23.6	MTF Edit =>Boxing.....	658
23.7	MTF Edit =>More.....	659
23.7.1	Smart Chainages in MTF Templates Grid.....	660
23.7.2	Smart Chainages in MTF Shift .....	661
23.7.3	Smart Chainages in MTF Specials Values Grid .....	662
23.7.4	Smart Chainages in MTF String Modifiers Grid .....	663
23.8	MTF Edit =>Settings .....	664
23.8.1	Smart Chainages in MTF Stripping Grid.....	665
23.8.2	MTF Edit =>Settings =>Design layer name.....	666
23.8.3	MTF Edit =>Settings =>Default chainage type.....	666
23.8.4	MTF Edit =>Settings =>Shape formation toggle .....	667
23.8.5	MTF Edit =>Settings =>Show Extra Start/end?.....	667
23.8.6	MTF Edit =>Settings =>Extra Start/end value .....	668
23.8.7	MTF Edit =>Settings =>Show extra start/end column? .....	668
23.8.8	MTF Edit =>Settings =>Show absolute column? .....	668
23.8.9	MTF Edit =>Settings =>Show interval column?.....	668
23.8.10	MTF Edit =>Settings =>Hinge link name .....	670
23.8.11	MTF Edit =>Settings =>Hinge widening treatment.....	670
23.9	MTF Edit =>Save.....	671
<b>24.</b>	<b>MTF Edit =&gt;Modify Left/Right .....</b>	<b>672</b>
24.1	MTF Links, Points, Sections, Strings and Trimeshes .....	674
24.1.1	MTF Links, Points and Strings .....	675
24.1.2	MTF Links and Layers .....	677
24.1.3	MTF Shapes and Trimeshes .....	678
24.2	Tick Boxes in the Grid .....	679
24.3	Turning Off Extra Start/End.....	679
24.4	Env Variable to Turn Off Interval Column.....	679
24.5	Env Variable to Turn Off Absolute Column.....	679
24.6	Layer and Link Under Command Name .....	680
24.7	New Look for MTF Modifier Panels .....	681
24.8	Alias for Left and Right Modifiers.....	682
24.9	Absolute Replaced by New Value Usage .....	683
24.10	Regions.....	684
24.11	Highlight Button.....	684
24.12	MTF Left/Right Modifiers Create Menu .....	685
24.12.1	Calculating Width, Height, Xfall or Slope from Original Definitions .....	685
24.12.2	Common Fields on Modifier Panels .....	687
24.12.3	Link or Link(s).....	688
24.12.4	Left/Right Modifiers => Create =>Fixed .....	689
24.12.4.1	Left/Right Modifiers => Create =>Fixed =>Insert .....	690
24.12.4.2	Left/Right Modifiers => Create =>Fixed =>Remove .....	698
24.12.4.3	Left/Right Modifiers => Create =>Fixed =>Trim .....	703
24.12.4.4	Left/Right Modifiers => Create =>Fixed =>Link.....	713
24.12.4.5	Left/Right Modifiers => Create =>Fixed =>Decisions .....	736

24.12.4.6	Left/Right Modifiers => Create =>Fixed =>from link	763
24.12.4.7	Left/Right Modifiers => Create =>Fixed =>to string	765
24.12.4.8	Left/Right Modifiers => Create =>Fixed =>to tin	772
24.12.4.9	Left/Right Modifiers => Create =>Fixed =>to RL	779
24.12.4.10	Left/Right Modifiers => Create =>Fixed =>to 2 Heights	785
24.12.4.11	Left/Right Modifiers => Create =>Fixed =>by 2 strings	787
24.12.4.12	Left/Right Modifiers => Create =>Fixed =>Interface to tin	793
24.12.4.13	Left/Right Modifiers => Create =>Fixed =>Boxing	795
24.12.5	Left/Right Modifiers => Create =>Template =>Cut	803
24.12.5.1	Left/Right Modifiers Cut =>Insert	803
24.12.5.2	Left/Right Modifiers Cut =>Insert cut template	805
24.12.5.3	Left/Right Modifiers Cut =>Link	806
24.12.5.4	Left/Right Modifiers Cut =>from link	810
24.12.5.5	Left/Right Modifiers Cut =>to string	813
24.12.5.6	Left/Right Modifiers Cut =>to tin	819
24.12.5.7	Left/Right Modifiers Cut =>to RL	826
24.12.5.8	Left/Right Modifiers Cut =>to 2 Heights	830
24.12.5.9	Left/Right Modifiers Cut =>by 2 strings	841
24.12.5.10	Left/Right Modifiers Cut =>Last slope	847
24.12.6	Left/Right Modifiers => Create =>Template =>Fill	849
24.12.7	Left/Right Modifiers => Create =>Interval	850
24.12.7.1	Left/Right Modifiers => Interval =>Alias	851
24.12.7.2	Left/Right Modifiers => Interval =>Interval	852
24.12.7.3	Left/Right Modifiers => Interval =>Interval Relative	853
24.12.7.4	Left/Right Modifiers => Interval =>Interval String TPs	854
24.12.8	Left/Right Modifiers => Create =>Snippet	855
24.12.9	Left/Right Modifiers - Creating Strings, Shapes and Tins	856
24.12.9.1	Left/Right Modifiers => Create =>Create =>Shapes	857
24.12.9.2	Left/Right Modifiers => Create =>Create =>Strings	860
24.12.9.3	Left/Right Modifiers => Create =>Create =>Tins	863
24.12.10	Left/Right Modifiers => Create =>Debug	866
24.12.11	Left/Right Modifiers => Create =>Pause	867
24.12.12	Left/Right Modifiers => Create =>Region	868
24.13	Defining and Using Snippets	869
24.13.1	What are MTF Snippets?	870
24.13.2	Creating a Snippet	871
24.13.3	An Example of a Snippet	872
24.13.4	How To Select a Snippet?	874
24.13.5	Comments in Snippets	877
24.13.6	Start and End Modes for a Snippet	878
24.13.7	Snippet Parameters	880
24.13.7.1	Snippet Parameter of Type REAL	881
24.13.7.2	Snippet Parameter of Type INTEGER	883
24.13.7.3	Snippet Parameter of Type TEXT	884
24.13.7.4	Snippet Parameter of Type SELECT	885
24.13.7.5	Snippet Parameter of Type CHOICE	887
24.13.7.6	Snippet Parameter of Type CHOICE2	889
24.13.7.7	Snippet Parameter of Type TICK	891
24.13.7.8	Snippet Parameter of Type COLOUR	893
24.13.7.9	Snippet Parameter of Type NAME	895
24.13.7.10	Snippet Parameters for Models	897
24.13.7.11	Snippet Parameters for Tins	904
24.13.7.12	Snippet Parameter of Type LAYER	911
24.13.7.13	Snippet Parameter of Type NAMED_GRADE	912
24.13.7.14	Snippet Parameter of Type INFO	913
24.13.7.15	Snippet Parameter of Type DISPLAY	914
24.13.7.16	Snippet Parameter of Type INCREMENT	916
24.13.7.17	Optional Parameters in Snippets	918

24.13.8 Automatic Parameters in Snippets.....	919
24.13.8.1 _AUTO_LR.....	919
24.13.8.2 _SCH.....	919
24.13.8.3 _ECH.....	920
24.13.8.4 _CL_REF.....	920
24.13.8.5 _CL_REF1.....	920
24.13.8.6 _CL_REF2.....	920
24.13.8.7 _CL_REF3.....	920
24.13.8.8 _CL_REF4.....	920
24.13.8.9 _NULL.....	921
24.13.8.10 _AUTO_0I, _AUTO_1J, ... , _AUTO_9R, _AUTO_AS, _AUTO_BT, ... _AUTO_HZ	921
24.13.8.11 _AUTO_05, _AUTO_16, _AUTO_27, ... _AUTO_49, _AUTO_AN, _AUTO_BO, ...	922
_AUTO_MZ.....	922
24.13.8.12 _INS_05, _INS_LR, _INS_0I.....	922
24.13.8.13 _INS_SIDE_05.....	923
24.13.8.14 _SIDE_FX.....	923
24.13.8.15 _AUTO_LAYER_LR.....	924
24.13.8.16 _APPLY_TIN.....	925
24.13.8.17 _APPLY_DESIGN_MODEL.....	925
24.13.8.18 _HINGE.....	925
24.13.8.19 _SIDE.....	925
24.13.8.20 _SILENT_NG.....	925
24.13.8.21 _PROJECT_ATTRIBUTE.....	926
24.13.8.22 Substrings of parameters.....	926
24.13.9 Arithmetic in Snippets.....	928
24.13.10 Trig and Maths Function Capabilities in Snippets.....	929
24.13.11 Snippet Directives.....	930
24.13.11.1 Deprecated C Preprocessor from V10.....	931
24.13.11.2 Directives for V11 Onwards.....	933
24.13.11.3 Flow control.....	934
24.13.11.4 Abbreviations.....	938
24.13.11.5 Comparisons of Data Types.....	939
24.13.11.6 Tokens and Tokens vs Tokens.....	941
24.13.11.7 Tokens vs Strings.....	958
24.13.11.8 Tokens vs Doubles.....	960
24.13.11.9 Tokens vs Integers.....	981
24.13.11.10 Values Defined and Value vs Value.....	988
24.13.11.11 Values vs Strings.....	993
24.13.11.12 Values vs Doubles.....	995
24.13.11.13 Values vs Integers.....	1012
24.13.12 Snippet Variables.....	1024
24.13.12.1 Defining Link Variables.....	1024
24.13.12.2 Defining General Variables.....	1025
24.13.12.3 Using Helper Functions.....	1026
24.13.12.4 Using Variables in Other Variables.....	1027
24.13.12.5 Using Variables in Commands.....	1028
24.13.13 Order of snippet processing.....	1029
24.13.14 Debugging Snippets.....	1035
24.13.14.1 Print Messages and Log Lines to the Output Window.....	1036
24.13.14.2 Intermediate Parsing Results.....	1038
24.13.14.3 Temporary MTF Snippet File.....	1039
24.13.15 Compiling Snippets.....	1041
24.13.16 Tips and Tricks.....	1042
24.13.17 Major Warning - You Will be Caught by This.....	1043
<b>25. Plot Menu.....</b>	<b>1044</b>
25.1 Zero Padding in Plot File Names.....	1044
25.2 New Plot to Models Node.....	1045

25.3	Extra Underscore in Long Sect Plot Names .....	1046
25.4	Title Blocks and External Data Sources .....	1046
25.4.1	Ad Hoc Queries .....	1046
25.4.1.1	EXCEL.....	1046
25.4.1.2	XML.....	1047
25.4.1.3	ODBC (Open Database Connectivity).....	1047
25.4.2	Drawing Register in Plot PPF's.....	1048
25.5	Plot Sheet .....	1050
25.5.1	Plot Sheet Create .....	1050
25.5.2	Plot Sheet Edit .....	1051
25.5.2.1	Plot Sheet Create/Edit.....	1052
25.5.3	Plot Sheet Plot .....	1063
25.5.4	Plot Sheet Delete .....	1064
<b>26.</b>	<b>Utilities Menu .....</b>	<b>1065</b>
26.1	Measure Bearing/Distance - Plan Area.....	1065
26.2	DZ in Xfall Between 2 Strings Inquire Panel.....	1066
26.3	Difference in Xfall Between 2 Strings Inquire (Advanced) Panel .....	1066
26.4	Macro Prototypes File.....	1067
26.5	Attribute Manipulator .....	1068
26.5.1	Create/Edit Attribute Manipulator File .....	1069
26.5.1.1	Attribute Manipulator Rules.....	1072
26.5.2	Apply Attribute Manipulator File.....	1077
26.6	Utilities =>A-G =>Explode .....	1079
26.7	Utilities =>H-Z =>Label Map .....	1080
<b>27</b>	<b>ADAC .....</b>	<b>1081</b>
27.1	ADAC.XML Overview .....	1082
27.1.1	ADAC XML Structure .....	1082
27.1.2	ADAC Assets .....	1086
27.1.3	ADAC Geometry Element .....	1087
27.1.4	Guidelines from an Authority Requesting ADAC.XML .....	1088
27.2	12d Approach to ADAC XML .....	1091
27.3	12d ADAC Workflow.....	1092
27.4	12d ADAC Menu.....	1095
27.4.1	Create Header .....	1096
27.4.2	Create ADAC Asset .....	1097
27.4.3	Edit ADAC Header/Asset.....	1100
27.4.3.1	ADAC Header Editor.....	1101
27.4.3.2	ADAC Asset Editor .....	1103
27.4.3.3	Common Features of ADAC Editors.....	1105
27.4.4	Validate.....	1115
27.4.5	Report .....	1118
27.4.6	Write ADAC XML File.....	1120
27.4.7	Import ADAC XML File.....	1121
27.4.8	ADAC Utilities.....	1124
27.4.8.1	ADAC Strings to Map File .....	1125
27.4.8.2	XSD to Map File.....	1128
27.4.8.3	XSD to Model.....	1130
27.4.8.4	Sync Geometry .....	1132
27.4.8.5	Create/Edit User Attributes to ADAC File .....	1133
27.4.8.6	Apply User Attributes to ADAC Elements.....	1137
27.4.8.7	ADAC XML File Editor .....	1138
27.4.9	User ADAC .....	1140
27.4.9.1	Setting Up a New ADAC Project .....	1141
27.4.9.2	Data Prep.....	1145
27.4.9.3	User ADAC Utilities.....	1171
27.4.9.4	12d 4.1 Chains .....	1189
27.4.9.5	Client Specific ADAC Design & Survey Menus.....	1193

27.5 ADAC Design and Survey Chains.....	1194
27.5.1 Survey to ADAC Chains.....	1195
27.5.1.1 ADAC Survey Base Chain pvf File .....	1195
27.5.1.2 ADAC Survey Base Chain.....	1197
27.5.2 Design to ADAC Chains.....	1205
27.5.2.1 ADAC Design Base Chain pvf File .....	1205
27.5.2.2 ADAC Design Base Chain.....	1207
27.6 Setting Up for ADAC.....	1216
27.6.1 Setting Up Map Files for ADAC .....	1217
27.6.1.1 Know What ADAC XML Is .....	1218
27.6.1.2 Know What Your Client Wants in the ADAC XML File.....	1220
27.6.1.3 How the ADAC Map File is Used .....	1221
27.6.1.4 Creating the ADAC Attributes for Map Files.....	1224
27.6.1.5 Notes On Creating the ADAC Attribute Structure .....	1229
27.6.1.6 Setting Up the Map File Selection Criteria .....	1229
27.6.1.7 More on Creating the ADAC Map Files .....	1247
27.6.2 What Data Prep is Needed for ADAC.....	1248
27.6.3 Setting up and Running ADAC Chains from the Menus.....	1249
27.6.3.1 Using the 12d Supplied Menu to Run ADAC Chains.....	1250
27.6.3.2 Setting Up Your User ADAC Menu .....	1252
27.6.4 Setting Up ADAC Templates .....	1258
27.6.4.1 ADAC Header Templates .....	1258
27.6.4.2 ADAC Asset Templates.....	1260
27.6.5 Setting Up Your User Keys .....	1262
<b>28 12d XML File Format .....</b>	<b>1263</b>
28.1 General Information about XML .....	1264
28.2 General Information about a 12d XML File .....	1266
28.3 Regularly Used Keyword Blocks.....	1267
28.3.1 Name.....	1267
28.3.2 Colour .....	1267
28.3.3 Line Style.....	1268
28.3.4 Chainage .....	1268
28.3.5 Weight.....	1268
28.3.6 Interval .....	1268
28.3.7 Time Created.....	1269
28.3.8 Time Updated .....	1269
28.3.9 Breakline.....	1269
28.3.10 Null .....	1270
28.3.11 Radius .....	1270
28.3.12 data_2d.....	1270
28.3.13 data_3d.....	1271
28.3.14 radius_data and major_data .....	1271
28.3.15 Available Transition Types.....	1273
28.4 Attributes.....	1274
28.5 Model .....	1276
28.6 Elements Contained in Models .....	1277
28.6.1 Tin.....	1278
28.6.2 Super Tin.....	1281
28.6.3 String Header Block.....	1283
28.6.4 Text Information .....	1285
28.6.4.1 Vertex Annotation Information.....	1285
28.6.4.2 Segment Annotation Information.....	1286
28.6.5 Arc String.....	1287
28.6.6 Circle String.....	1289
28.6.7 Drainage String.....	1290
28.6.8 Feature String.....	1295
28.6.9 Plot Frame String.....	1296
28.6.10 Super String .....	1299

28.6.10.1	Defining the Coordinates of the Vertices .....	1302
28.6.10.2	Geometry of the Horizontal Segments.....	1303
28.6.10.3	Colour .....	1308
28.6.10.4	String, Vertex and Segment Attributes .....	1309
28.6.10.5	Vertex Id's (Point Id's) .....	1312
28.6.10.6	Symbols at Vertices .....	1313
28.6.10.7	Tinability.....	1315
28.6.10.8	Round or Box (Culvert) Pipes .....	1316
28.6.10.9	Vertex Text and Vertex Annotation.....	1318
28.6.10.10	Segment Text and Segment Annotation .....	1320
28.6.11	Super Alignment String .....	1322
28.6.11.1	Horizontal Data Block .....	1327
28.6.11.2	Horizontal_Parts When Geometry is Defined by IP Method Only .....	1329
28.6.11.3	Vertical Data Block .....	1334
28.6.11.4	Geometry of the Vertical Segments.....	1336
28.6.11.5	Vertical_parts When VG is Defined by IP Method Only .....	1339
28.6.12	Text String .....	1344
28.6.13	Trimesh.....	1346

# 1. Moving to 12d Model 11

## 1.1 About This Manual

This manual contains information on:

- (a) what you need to do when installing and moving across to **12d Model 11**.

Sections in this chapter contain information about:

[1.3 Dongles for 12d Model 11](#)

[1.4 New Installers and Uninstallers](#)

[1.5 User, User\\_Lib and env.4d for 12d Model 11](#)

[1.7 Network Dongles and 12d Model 11](#)

- (b) details on some of the new, or modified, options in **12d Model 11**.

The chapters [2. New Front Panel](#) to [12. Bits and Pieces](#) contain general information.

The chapters [13. Menus on Views](#) to [26. Utilities Menu](#) are set out in the order that options appear on the **12d Model 11 menus**, **Reference** manual and Context Sensitive **Help**.

Chapter [27 ADAC](#) contains a description of the IPWEA Standard, ADAC.XML, and full information on how it is implemented within **12d Model 11**.

All of the relevant material in this manual is also in the **12d Model 11 Reference** manual and the Context Sensitive **Help**.

But first you need to read the section [1.2 Some Important Notes on 12d Model 11](#).

## 1.2 Some Important Notes on *12d Model 11*

1. **12d Model 11** will:

(a) NOT RUN on Windows Vista, XP or earlier versions of Windows.

(b) **12d Model 11** WILL RUN on Windows 7

(c) **12d Model 11** MAY RUN on Windows 8.1

We are running **12d Model 11** on Windows 8.1 and most options run. However there are still unresolved problems in Windows 8.1 itself and so we can't be certain that everything runs as expected.

2. Hardlock dongles will NOT work with **12d Model 11**.

If you have a Hardlock dongle, please contact your Reseller.

3. Wibu dongles need drivers of at least version 6.1

To obtain new drivers, see [1.4 New Installers and Uninstallers](#).

you have a Hardlock dongle, please contact your Reseller.

4. There is a new **dongles.4d** file that replaces the WIBU network Environment variables

The WIBU network environment variables WIBU\_DONGLE\_4D and WIBU\_IPADDR are superceded by the new **dongles.4d** file.

**However** the **dongles.4d** file shipped with **12d Model** contains no references to network dongles and in that case, the WIBU\_DONGLE\_4D and WIBU\_IPADDR environment variables are still used.

So apart from needing to copy your V10 env.4d file to your V11 User, people using Wibu networks dongles **will not need to do anything** for V11 to work as before **until** they start using Wibu **Codemeter Containers**. See [1.5.3 env.4d](#).

5. **12d Model 11** projects CAN NOT be opened in earlier versions.

Although there are many new database objects **12d Model 11**, any project from an earlier version will open up in **12d Model 11** and a save will write the project out in **12d Model 11** format. However **12d Model 11** projects CAN NOT be opened in earlier versions of **12d Model**.

6. The new **12d Model 11 Uninstaller** deletes everything in the 11.00 Area

The **12d Model 11** Uninstallers **delete all files** in the 32-bit or 64-bit folders

**Program Files\12d\12dmodel\11.00**

or

**Program Files (x86)\12d\12dmodel\11.00**

So **do not** add or modify any files in those areas because they will be deleted by the next Uninstall (see [1.4.3 Uninstallers for 12d Model](#)).

Continue to [1.3 Dongles for 12d Model 11](#) or return to [1.1 About This Manual](#).

## 1.3 Dongles for 12d Model 11

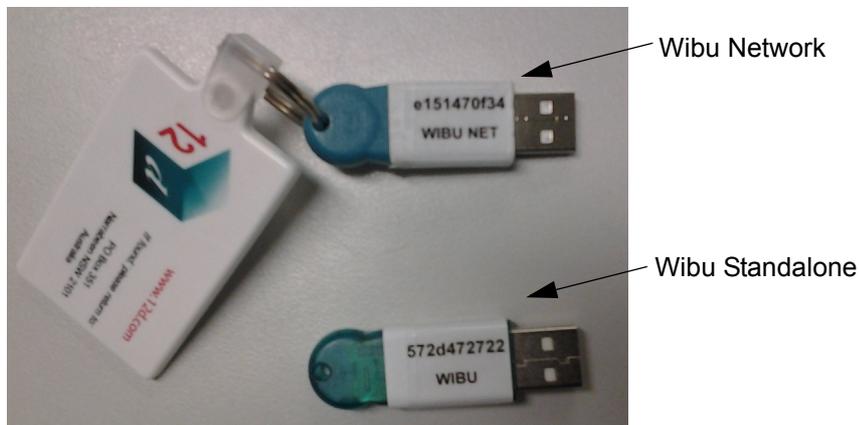
**12d Model 11** will NOT WORK with **Hardlock** dongles. If you have a **Hardlock** dongle, please contact your **12d Model** Reseller.

**12d Model 11** will work with any Wibu stand alone and Wibu network dongles that you already have (these are the dongles that we have been shipping for the last decade) but as from January 2015, there will also be available the new type of lock from Wibu, the Wibu **CodeMeter Container**.

**12d Model 11** will only work with the following physical hardware locks:

(a) **Existing Wibu dongles**

**Wibu Standalone dongles** (12d dongle number starting with 572d) or **Wibu Network dongles** (12d dongle number starting with e151)

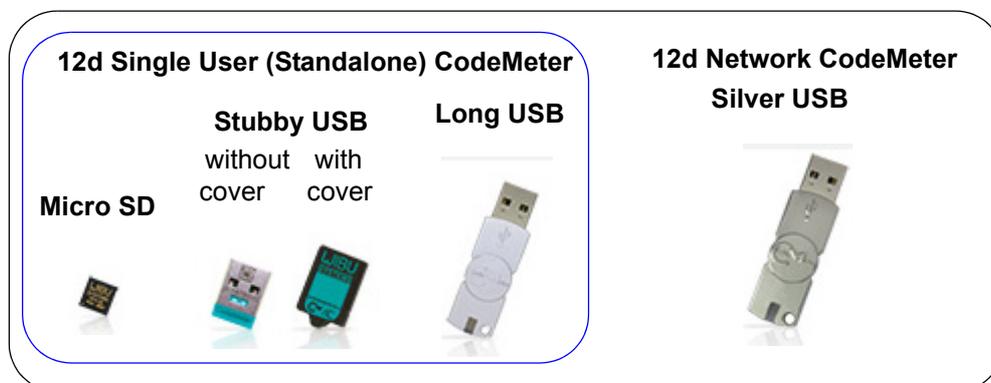


The **Wibu drivers** for **12d Model 11** must be at least version **6.1**. See [1.4 New Installers and Uninstallers](#)

or, new for V11,

(b) **Wibu Codemeter Containers**

The **Wibu Codemeter Containers** are similar to the earlier Wibu standalone and network dongles except that they come in a wider variety of hardware shapes some of which will be more suitable for portable and tablet computers



We will refer to the *Wibu CodeMeter Containers* as **CodeMeter Containers** or **CodeMeters**.

**CodeMeters** have *12d dongle numbers* starting with 5c2d.

Codemeter Containers as standalone dongles can be used with **12d Model 10 C1n** onwards but will not work with earlier versions of **12d Model**.

As network dongles, Codemeter Containers can NOT be used for **12d Model 10** but only be **12d Model 11** onwards.

So if you move to a **CodeMeter Container**, you won't be able to run the **12d Model 9** and **earlier** 12d.exe's. But of course you can always open projects from those versions in **12d Model 11**.

**Important Note for 12d Field users:**

In **12d Model 11**, the **12d Field modules** will only work with **CodeMeter Containers**.

If you are a **12d Field** user, your **12d Model** Reseller will probably have already contacted you but if not, please contact them when you are preparing to upgrade to **12d Model 11**.

**Important Note**

Do not attach a dongle to your computer until after the appropriate driver is installed.

Continue to [1.4 New Installers and Uninstallers](#) or return to [1.1 About This Manual](#).

## 1.4 New Installers and Uninstallers

See

[1.5 User, User\\_Lib and env.4d for 12d Model 11](#)

[1.4.2 Installers for 12d Model](#)

[1.4.3 Uninstallers for 12d Model](#)

Or return to [1.1 About This Manual](#)

### 1.4.1 Installers for Wibu and CodeMeter Drivers

There is a new installer for installing the drivers required for **Wibu dongles** and/or **CodeMeter Containers**.



You simply select the drivers to match your dongle type - Wibu and/or CodeMeter.

#### Important Notes

1. Do not attach a dongle to your computer until after the appropriate driver is installed
2. The Wibu Dongle drivers need to be at least version **6.1**.

If your computer already has Wibu drivers installed and when you start **12d Model 11** it complains that the **Wibu** dongle drives are NOT at least version **6.1** then

- (a) Uninstall the existing Wibu drivers
- (b) Install new Wibu drivers from the 12d Driver installer or download and the latest Wibu drivers from the **12d** web site, [www.12d.com](http://www.12d.com).

Continue to [1.4.2 Installers for 12d Model](#) or return to [1.4 New Installers and Uninstallers](#).

## 1.4.2 Installers for 12d Model

There are new **Installers** for installing the 32-bit and the 64-bit **12d Model 11**.

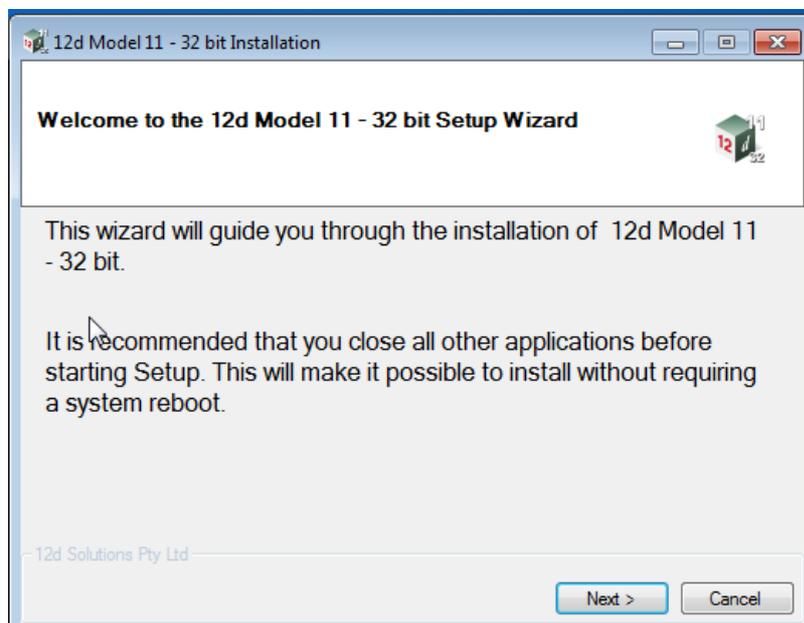
The Installers are much faster to install than the Installers for **12d Model 10** and it is much easier to do a silent install across a network with the new Installers.

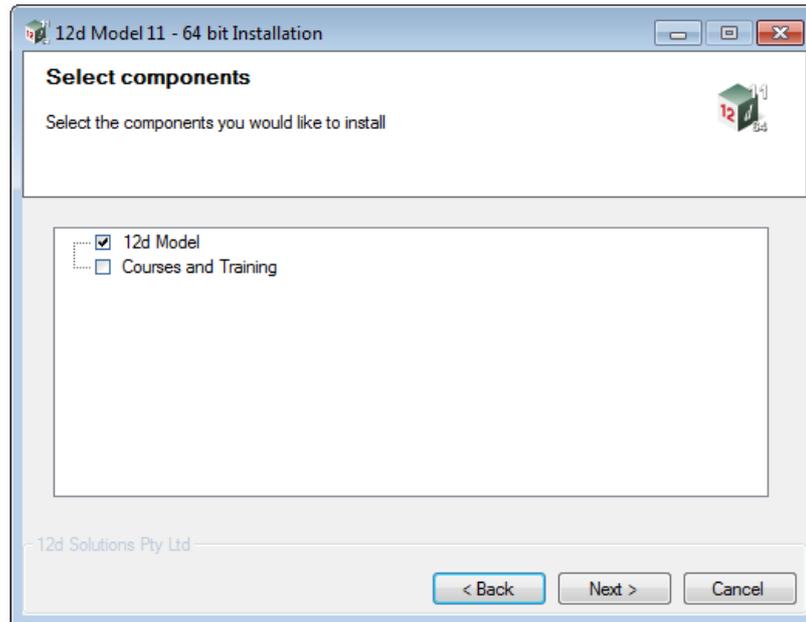
With the different libraries required for the latest Microsoft compiler we are using, there is no **12d Model 11** for Windows XP.

So there will be two different Installers

1. Installer for 32-bit **12d Model** exe for Windows 7.
2. Installer for 64-bit **12d Model** exe for Windows 7.

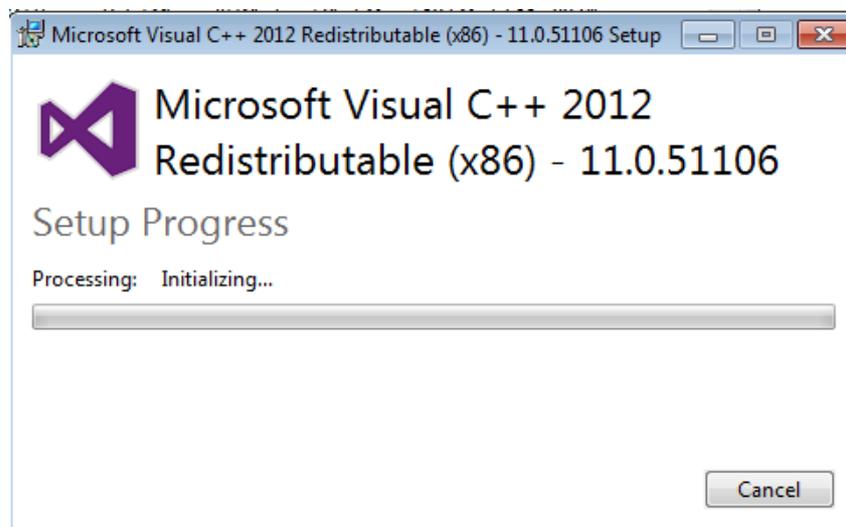
After starting the Installer, panels come up to guide you through the process.





When the Select components panel appears, tick **12d Model** and if you want the Courses and Training material, tick **Courses and Training**.

During the Install, the Microsoft Visual C++ 2012 Redistributable panel will appear.



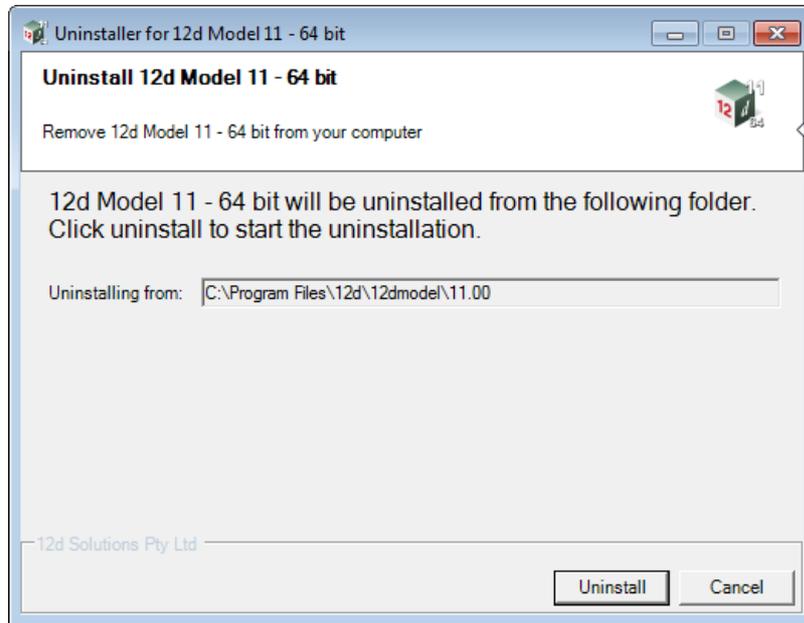
Do not click on anything, especially Cancel. The panel will disappear by itself.

Continue to [1.4.3 Uninstallers for 12d Model](#) or return to [1.4 New Installers and Uninstallers](#).

### 1.4.3 Uninstallers for 12d Model

There are new **Uninstallers** for uninstalling the 32-bit and the 64-bit **12d Model 11**.

The uninstallers are much faster to install than the uninstallers for **12d Model 10** and it is much easier to do a silent uninstall across a network with the new uninstallers.



**BUT** unlike the previous Uninstallers, the new ones **delete all files** in the 32-bit or 64-bit folders  
**Program Files\12d\12dmodel\11.00**

and

**Program Files (x86)\12d\12dmodel\11.00**

So **do not** add or modify any files in those areas because they will be deleted by the next Uninstall.

So make sure that in **User** you keep any modifications of the files in **Set\_ups**, and any of your own library files in **User\_Lib**.

Continue to [1.5 User, User\\_Lib and env.4d for 12d Model 11](#) or return to [1.4 New Installers and Uninstallers](#).

## 1.5 User, User\_Lib and env.4d for *12d Model 11*

See

[1.5.1 User](#)

[1.5.2 User\\_Lib](#)

[1.5.3 env.4d](#)

[1.5.4 nodes.4d](#)

Or continue to [1.7 Network Dongles and 12d Model 11](#) or return to [1.1 About This Manual](#)

### 1.5.1 User

**User** contains the files that you have modified files and are to be used instead of the files of the same name in **Set\_ups** (e.g. names.4d, linestyl.4d).

No **User** folder is created when *12d Model 11* is installed but if you had files in your **User** for *12d Model 10* then you will probably also want them to be used in *12d Model 11*.

If this is the case you will need to create a **User** folder for **V11**.

In **V10**, the default place for **User** is `c:\12d\10.00\User` but it could have an entirely different location and name, and then the pathname to it is given by the V10 environment variable **USER\_4D**.

In **V11**, the default place for **User** is `c:\12d\11.00\User` or it could be pointed to by the V11 environment variable **USER\_4D**.

So if you require a **User** folder in V11, you need to create the **User** folder in the correct location and copy the files from V10 **User** that you want to use for V11 to the new V11 **User** location.

#### **A Note on a Problem When NOT Using the Default Location for User.**

If you are going to use the environment variable **USER\_4D** to point to the location of **User**, then you need to update the value of `USER_4D` in the `env.4d` file to be the pathname to **User**.

Unfortunately a “chicken or the egg” situation exists here because the value for `USER_4D` is usually given inside the file `env.4d` which is in the **User** pointed to by `USER_4D`.

There are ways around this but if you leave **User** in a default location then you don't have to worry about this conundrum.

Continue to [1.5.2 User\\_Lib](#) or return to [1.5 User, User\\_Lib and env.4d for 12d Model 11](#).

## 1.5.2 User\_Lib

**User\_Lib** contains your own library files.

No **User\_Lib** folder is created when **12d Model 11** is installed but if you had library files in your **User\_Lib** for **12d Model 10** then you will probably also want them to be used in **12d Model 11**.

If this is the case you will need to create a **User\_Lib** folder for **V11**.

In **V10**, the default place for **User\_Lib** is **c:\12d\10.00\User\_Lib** but it could have an entirely different location and name, and then the pathname to it is given by the V10 environment variable **USER\_LIB\_4D**.

In **V11**, the default place for **User\_Lib** is **c:\12d\11.00\User\_Lib**, or it could be pointed to by the V11 environment variable **USER\_LIB\_4D**.

So if you require a **User\_Lib** folder in V11, you need to create the **User\_Lib** folder in the correct location and copy the files from V10 **User\_Lib** that you want to use for V11 to the new V11 **User\_Lib** location.

If in V11 you are going to use the V11 environment variable **USER\_LIB\_4D** to point to the location of **User**, then you need to update the value of **USER\_LIB\_4D** in the **env.4d** file for V11 to be the pathname to **User\_Lib**.

Continue to [1.5.3 env.4d](#) or return to [1.5 User, User\\_Lib and env.4d for 12d Model 11](#).

### 1.5.3 env.4d

**env.4d** is the file that contain the values you want for any of the **12d Model** environment variables. The default location for the file **env.4d** is in the **User** folder.

A default *env.4d* file is installed with the V11 **Set\_Ups** folder but if you were using your own *env.4d* file in V10 then you will probably want to also use your own *env.4d* file in V11.

If this is the case you will need to make sure that you have copied your **env.4d** file from your V10 **User** folder to the V11 **User**. See [1.5.1 User](#).

Continue to [1.5.4 nodes.4d](#) or return to [1.5 User, User\\_Lib and env.4d for 12d Model 11](#).

### 1.5.4 nodes.4d

A new **nodes.4d** for **12d Model 11** will be emailed to you by your **12d Model 11** Reseller along with instructions on how to install it.

Note that the **nodes.4d** for V11 is now an XML format and so can not be easily edited. For information on editing and modifying, and reporting on the **nodes.4d** file, please see [1.6 Nodes.4d is XML](#).

Continue to [1.6 Nodes.4d is XML](#) or return to [1.5 User, User\\_Lib and env.4d for 12d Model 11](#), [1.5.4 nodes.4d](#) or [1.1 About This Manual](#).

## 1.6 Nodes.4d is XML

The **nodes.4d** for V11 is now an XML format and so can not be easily edited, or even viewed, in its native XML format.

So the program that is used to install the **nodes.4d** file has been upgraded so that it can now:

- (a) Install a nodes.4d as before
- but also load an existing XML nodes.4d and
- (b) load another nodes.4d
- (c) look at all the entries in the file and display information on each entry (start & end dates, modules authorised etc)
- (d) move entries up and down
- (e) delete entries
- (f) create an html report on the entries
- (g) create a new nodes.4d file from selected entries

This program is accessible from inside **12d Model** by

- (a) the **Nodes** button on the Front screen (the one before you select a project)
- (b) the option **Projects =>Management =>Dongles =>Nodes.4d editor**  
and also from outside 12d Model
- (c) the program is called 12dNodesUtility.exe and is installed as a 32-bit program in  
**Program Files (x86) \12d\Nodes\11.0**

For information on the nodes.4d editor, see [14.5.3.6 Nodes.4d Editor](#).

Continue to [1.7 Network Dongles and 12d Model 11](#) or return to [1.5 User, User\\_Lib and env.4d for 12d Model 11](#) [1.5.4 nodes.4d](#) or [1.1 About This Manual](#).

## 1.7 Network Dongles and *12d Model 11*

If you are using a **12d** network dongle with **12d Model 10** then the V10 environment variable `WIBU_DONGLE_4D` and possibly `WIBU_IPADDR` were required to tell the system to use a network dongle instead of a stand alone dongle, and possibly which computer or IP address the dongle is on. These V10 environment variables are in the V10 `env.4d` file.

So if you are wanting to use the network dongle after **12d Model 11** is installed, you definitely need to copy your `env.4d` file from your V10 User to your V11 User. See [1.5.1 User](#).

Or you need to edit the V11 `env.4d` file and set the correct values for `WIBU_DONGLE_4D` and possibly `WIBU_IPADDR`.

Otherwise **12d Model 11** will not see the network dongle and be able to be authorised.

Return to [1.1 About This Manual](#).

# 2. New Front Panel

See

- [2.1 New Front Panel](#)
- [2.2 Browse Button and Open Tab](#)
- [2.3 New Button and New Tab](#)

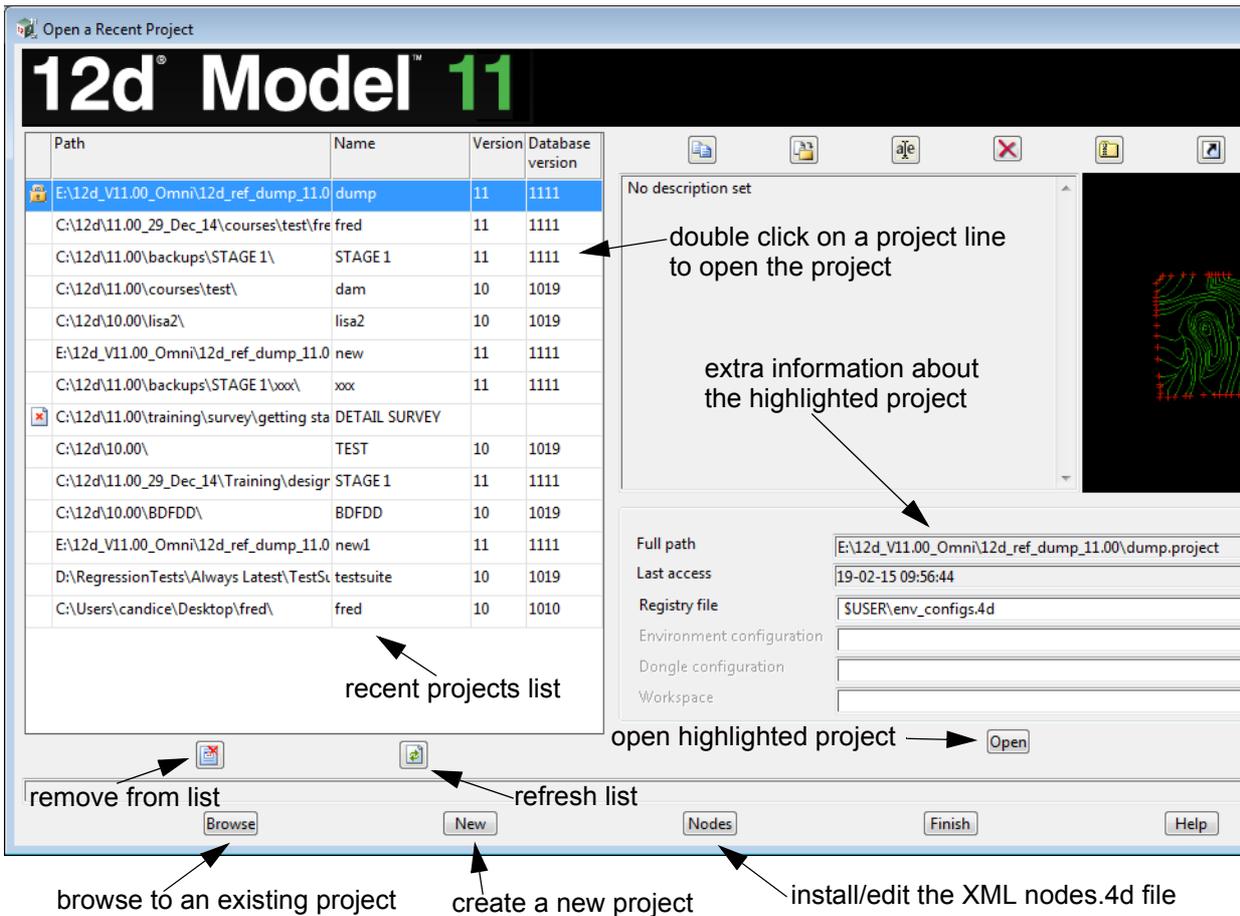
## 2.1 New Front Panel

The Front Panel has been modified so that it will fit on a Tablet with resolution 1200 x 600.

The Recent Project list is displayed on the left of the panel, and double clicking on a project in that list will open the project.

Extra Information such as full path, latest address date, Registry file, *etc.* is displayed for the highlighted project. Clicking on the **Open** button will open the highlighted project.

The icons have been rearranged - the **Remove from list** and **Refresh** icons are under the Recent Projects list.



For information on the *Browse* button, go to [2.2 Browse Button and Open Tab](#)

For information on the *New* button, go to [2.3 New Button and New Tab](#)

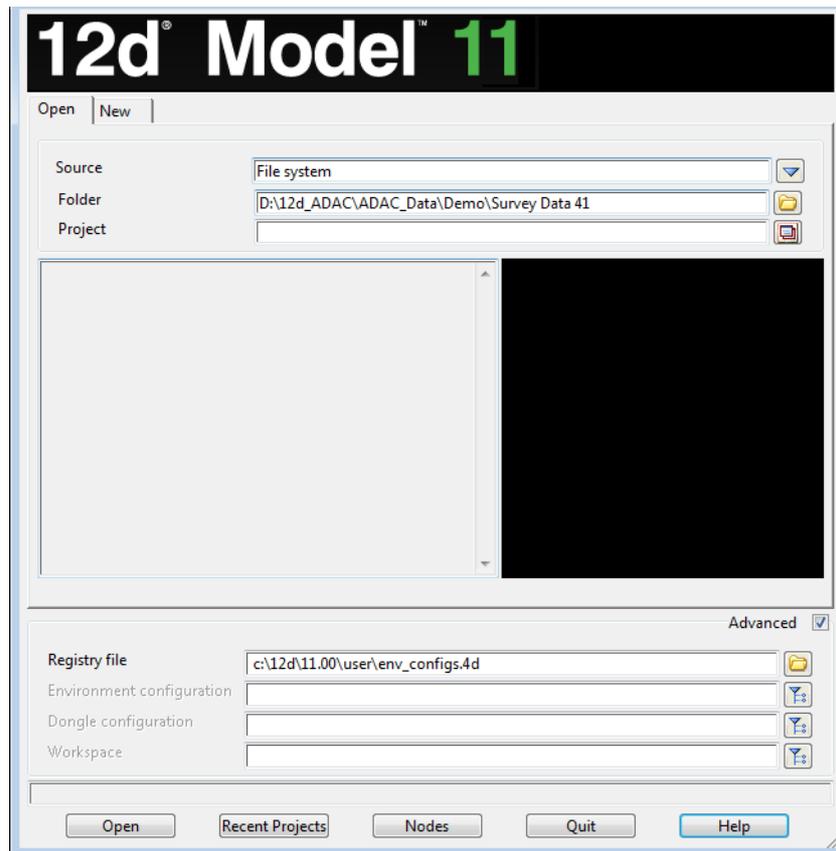
## 2.2 Browse Button and Open Tab

Clicking on the **Browse** button brings up the **Open** tab that allows you to browse and open an **existing** project. This is used when opening a project that is not in the *Recent Projects* list.

The **Open** button opens the project that has been browsed to.

The **Recent Projects** button takes you back to the original Front Screen.

The **New** tab takes you to the tab for creating a new project.



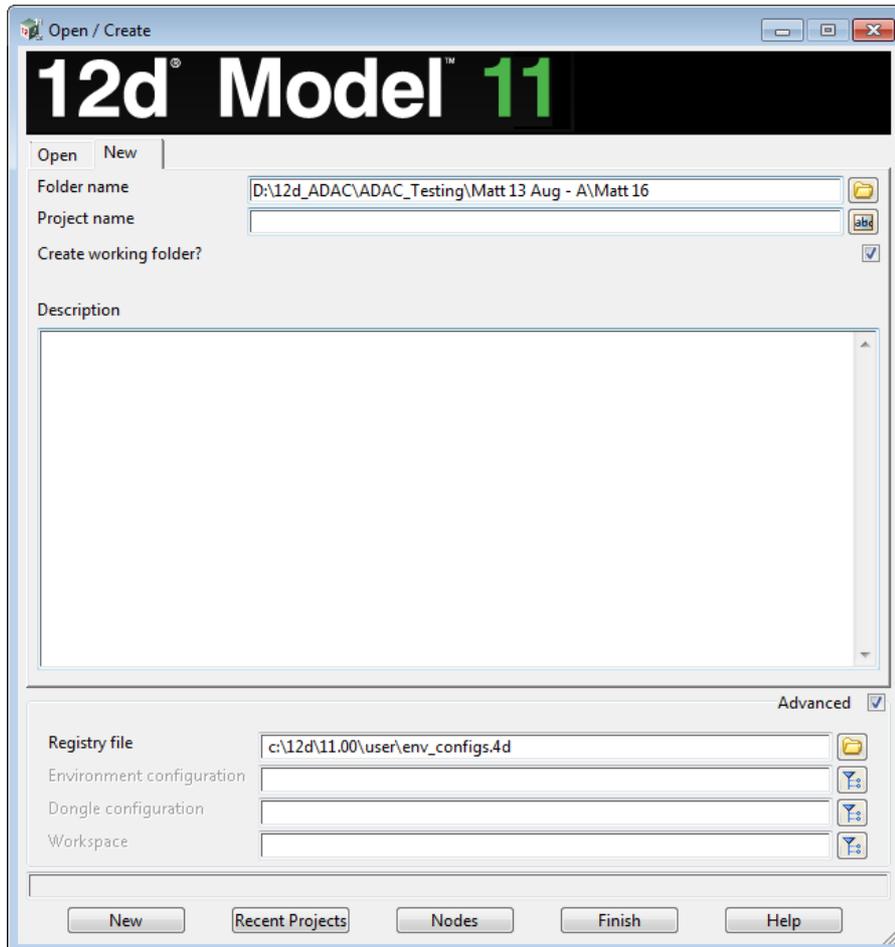
## 2.3 New Button and New Tab

Clicking on the **New** button brings up the **New** tab that allows you to create a **new** project.

The **New** button creates the new project.

The **Recent Projects** button takes you back to the original Front Screen.

The **Open** tab takes you to the tab for browsing to open an existing project.



# 3 New Objects

See

[3.1 Trimeshes](#)

[3.2 Dimensions](#)

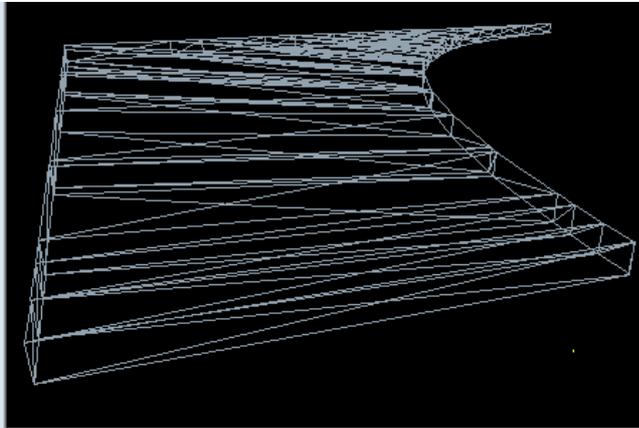
[3.3 Leaders](#)

[3.4 Tables](#)

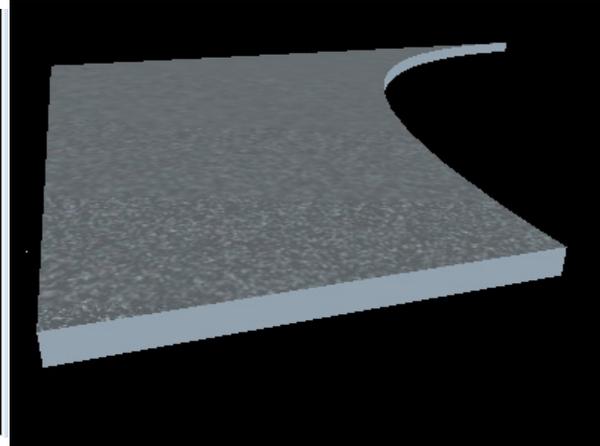
## 3.1 Trimeshes

To model a continuous 3D surface, trimeshes have been introduced.

A trimesh is a web of connected 3D triangles. where none of the triangles intersect each other.



Trimesh - not shaded



Trimesh - shaded

A trimesh can have a name of up to two hundred alphanumeric characters and spaces. The trimesh name can be blank.

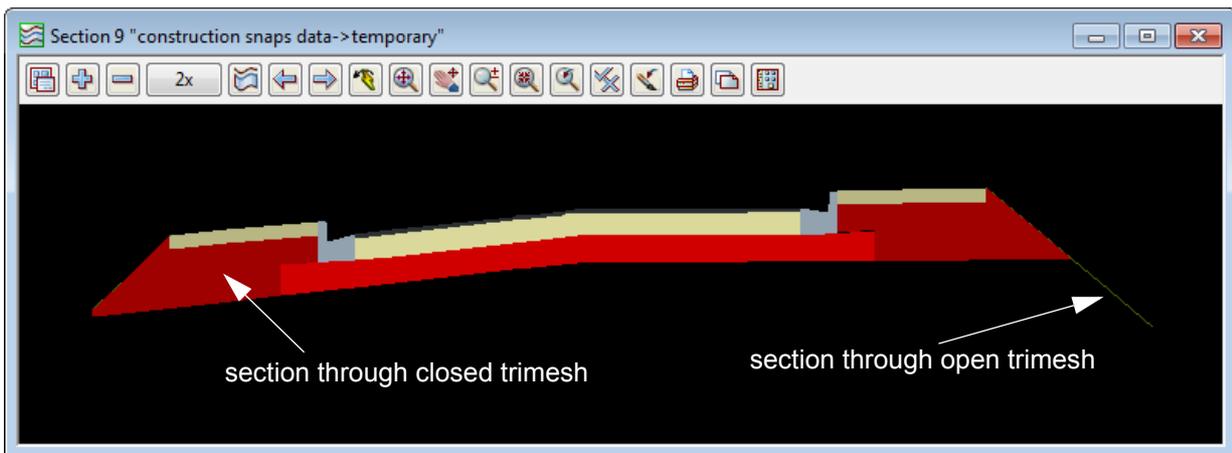
There is a default colour for the triangles (faces) in a trimesh but individual faces can be also be given a different colour to the default colour.

A trimesh has a **surface area** which is the sum of the area of each of the triangles in the trimesh.

A trimesh is **closed** if for each side of a triangle in the trimesh, there is another triangle in the trimesh butting up to it. A **closed** trimesh can have a **volume**.

If a trimesh is not **closed** then it is said to be **open**. An **open** trimesh **does not** have a **volume**.

When taking a section through a trimesh, the **outline** of the trimesh is shown and if the trimesh is **closed**, then the trimesh is **colour filled**.



A trimesh is a generalisation of a tin and unlike a tin, a trimesh can fold under itself.

There are numerous tools in **12d Model** for generating **trimeshes** for representing pavement layers, gutters, tunnels etc as 3D objects.



## 3.2 Dimensions

Various types of **Dimensions** can now be created with the **Dimension** options.

The image shows the 'Cad Dimension' toolbar at the top, which contains various icons for dimensioning. Below it is the 'Dimension' dropdown menu, which lists several options. Two sub-menus are highlighted with circles: one for 'Horizontal', 'Vertical', 'Aligned', and 'Rotated'; and another for 'Horizontal segment', 'Vertical segment', 'Aligned segment', and 'Rotated segment'. Arrows point from these sub-menus to descriptive text on the right.

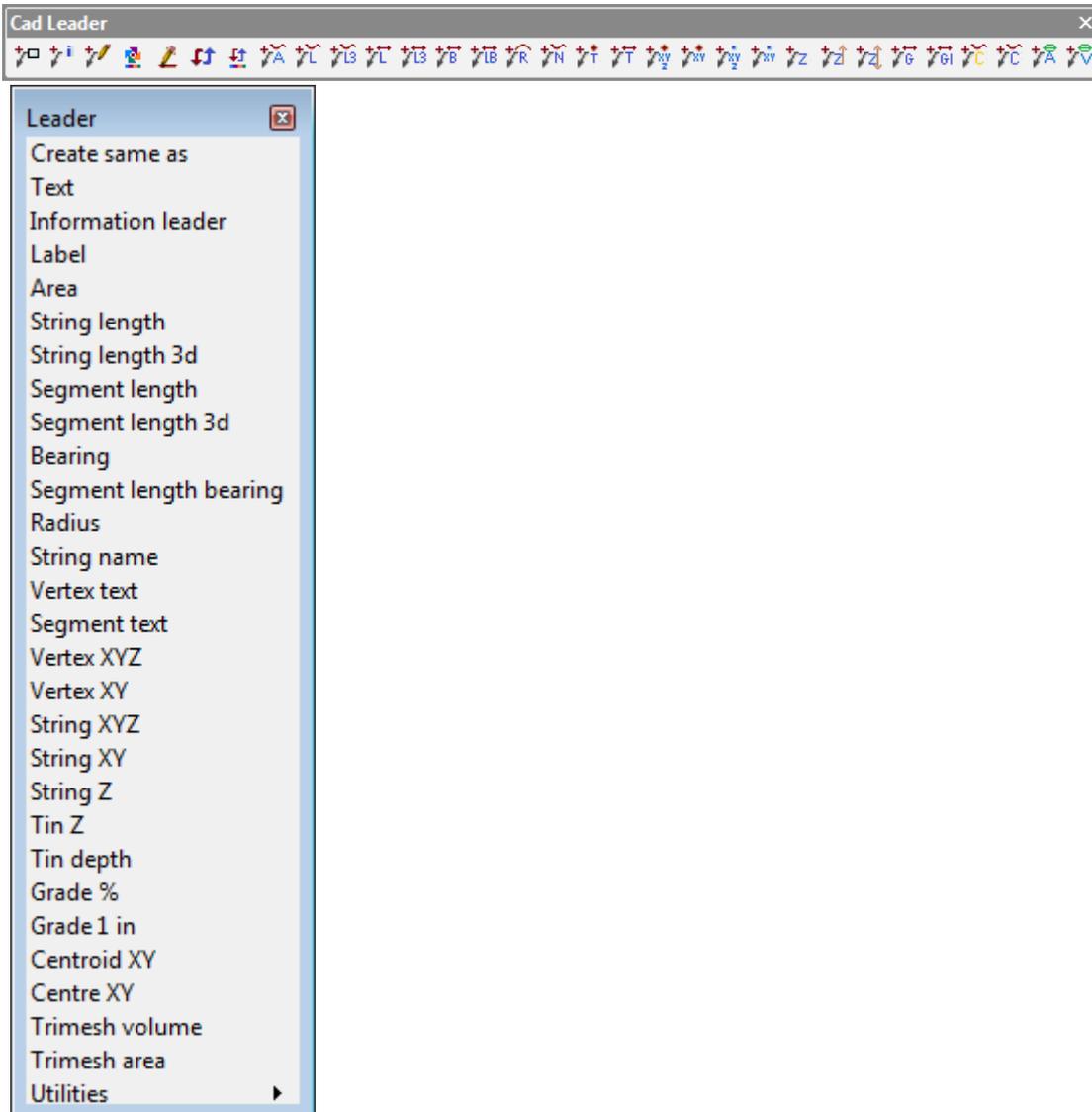
Dimension Option	Description
Create same as	create same dimension type as a selected dimension
Linear	various 2D distances
Angle 2 lines	angle between 2 lines
Angle 3 points	
Angle arc	
Length	2D length of a segment
Drop segment	point drop onto segment
Drop string	point drop onto string
Area	area of a polygon
Radial	
Diameter	
Jogged radius	
<b>Special Linear - Two Points Defines the Extent</b>	
Horizontal	delta x - i.e. Rotation of 0. Two points defines the extents.
Vertical	delta y - i.e. Rotation of 90. Two points define the extent
Aligned	Rotation given by 2 pts. Two points define the extent
Rotated	User given rotation. Two points define the extent.
<b>Special Linear - Segment Defines the Extent</b>	
Horizontal segment	delta x - i.e. Rotation of 0. Segment defines the extents.
Vertical segment	delta y - i.e. Rotation of 90. Segment defines the extent
Aligned segment	Rotation given by segment. Segment defines the extent
Rotated segment	User given rotation. Segment defines the extent.
Utilities	various options to work with dimensions.

Each type of **Dimension** remembers the items it was created from (Association). In most cases when the items are modified, the **Dimensions** and the associated values on the Dimension dynamically change.

See [18.1 CAD Dimension](#).

## 3.3 Leaders

Various types of **Leaders** can now be created with the **Leader** options

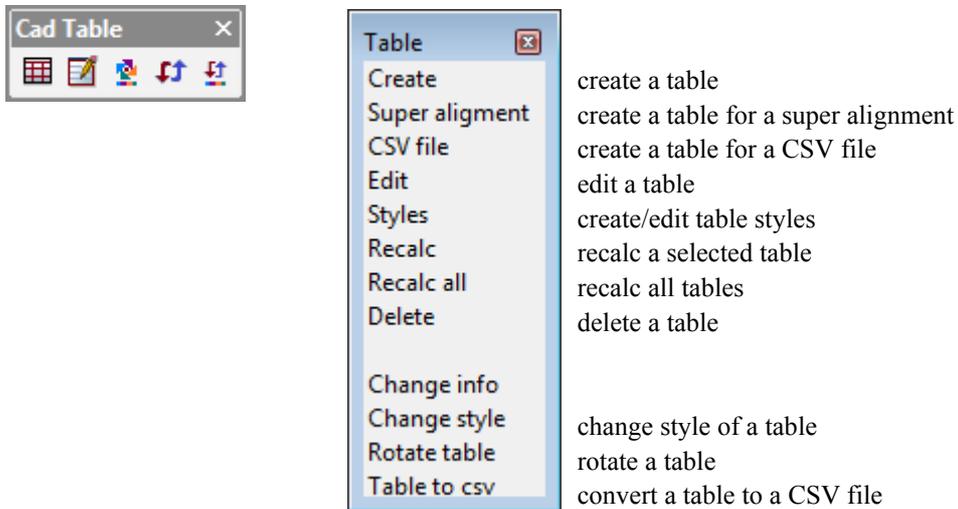


Apart from the Trimesh Volume and Trimesh area leaders, each type of Leader can remember the items it was created from (Association). In most cases when the items are modified, the Leader and the associated values from the items dynamically change. See [18.2.1 Leader Association](#).

See [18.2 CAD Leader](#).

## 3.4 Tables

There is a new **CAD Table** toolbar and **CAD =>Table** menu.



The **CAD Table** options create and edit tables.

There are options to create some tables from super alignments and they are associated with the super alignment so they can automatically update with changes in the super alignment.

See [18.4 CAD Table](#).

# 4. Special File Types

See

[4.1 12daz - Zipped .12da File](#)

[4.2 Label Map File](#)

[4.3 Map File](#)

[4.4 Names.4d File](#)

## 4.1 12daz - Zipped .12da File

There is a new file type **12daz** with the ending **.12daz**.

A **12daz** file is a **zipped 12da** file.

*Zipping the 12da* file can result in files a twentieth of the size of the original 12da file (*i.e.* 5% of the original), which makes it much better for emailing around, archiving, *etc.*

The **Write 12d Solutions Archive Data** panel, which replaces the **Write 12d Solutions Ascii Data** panel, creates **12daz** files as one of its **Format** choices.

The **Read 12d Solutions Archive Data** panel, which replaces the **Read 12d Solutions Ascii Data** panel, recognises and reads **12daz** files.

## 4.2 Label Map File

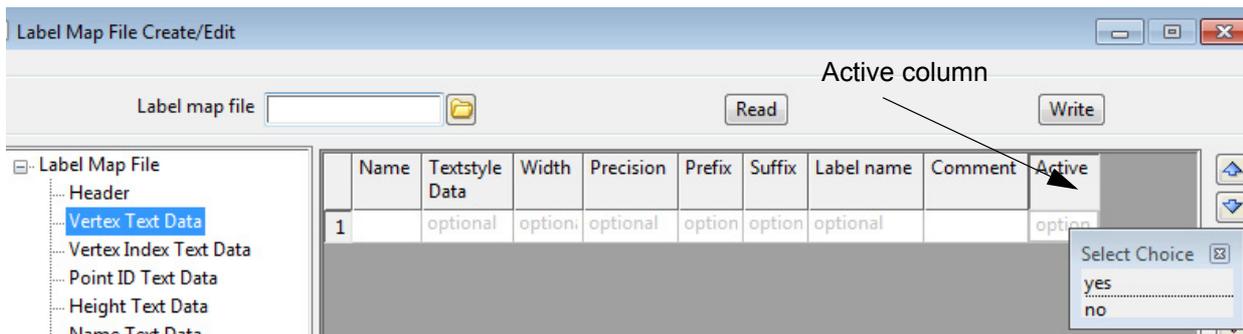
### 4.2.1 Active Column in Grids

There is now an **optional Active** column in each grid in the **Label Map File** so that the individual rows of the grid can be turned off and so not used in the **Label Map File**. This replaces having to delete the row.

If the value in the Active column for a row column is left **blank**, then the row is used.

If the value in the Active column for a row column is **yes**, then the row is used.

If the value in the Active column for a row column is **no**, then the row is **not** used.



## 4.3 Map File

See

[4.3.1 Active Column in Grids](#)

[4.3.2 Attributes Node](#)

[4.3.3 Group Column in Grids](#)

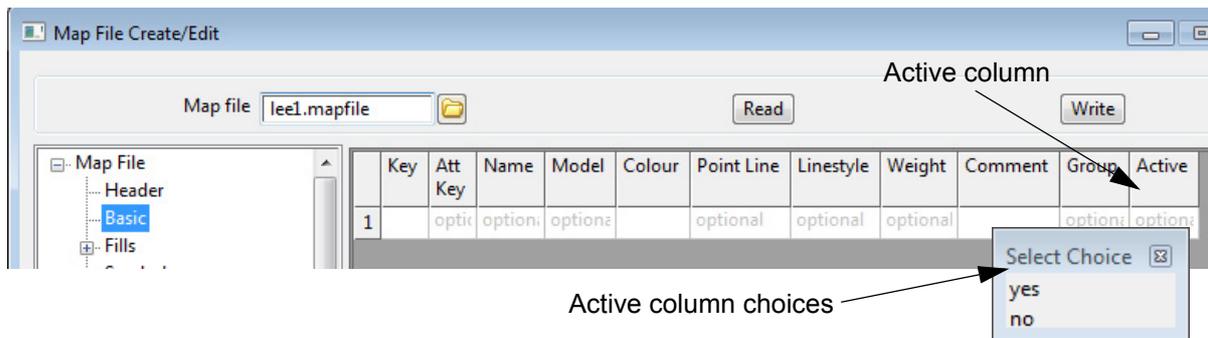
### 4.3.1 Active Column in Grids

There is now an **optional Active** column in each grid in the **Map File** so that the individual rows of the grid can be turned off and so not used in the Map File. This replaces having to delete the row.

If the value in the Active column for a row column is left **blank**, then the row is used.

If the value in the Active column for a row column is **yes**, then the row is used.

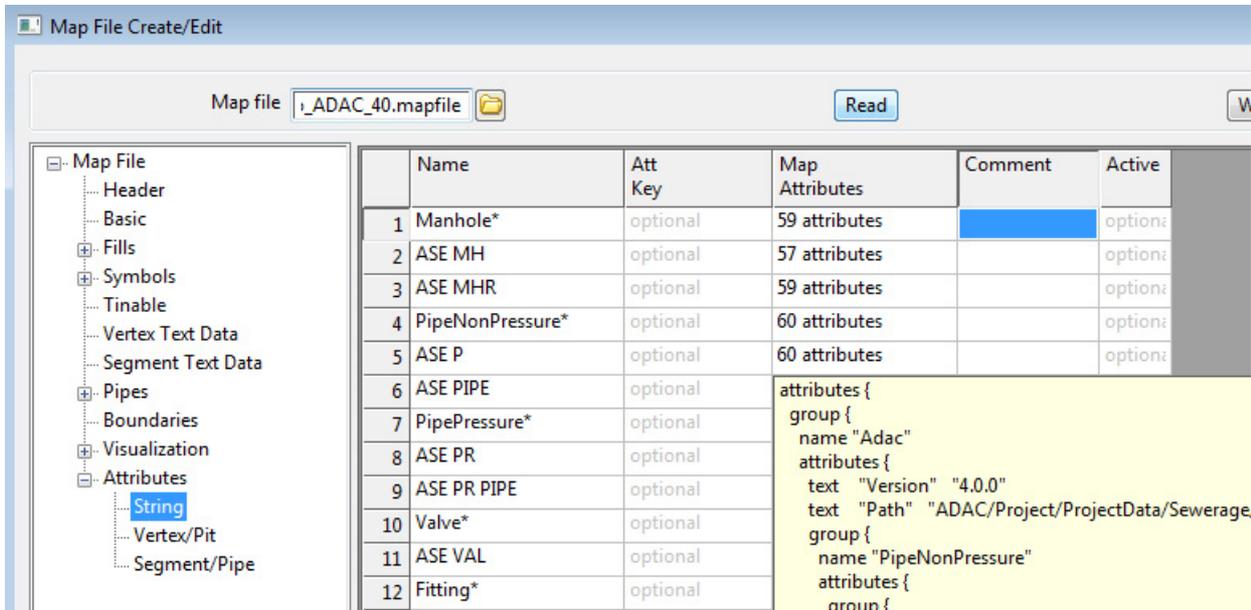
If the value in the Active column for a row column is **no**, then the row is **not** used.



### 4.3.2 Attributes Node

There is a new **Attributes** node in the **Map File** which is used to apply attributes to Strings, Vertices and/or Segments.

#### Attributes >Strings



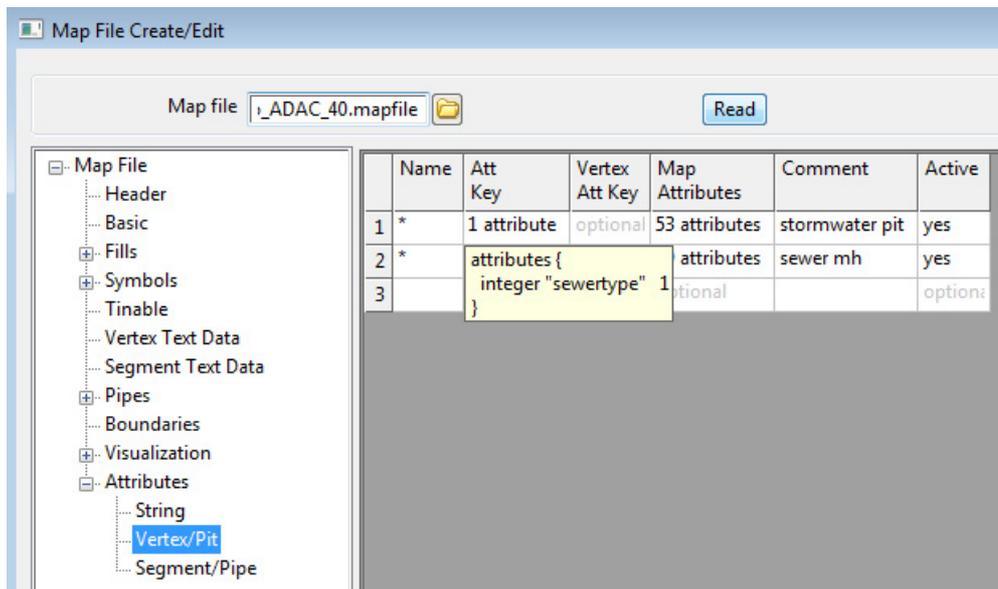
When a string finds a match in the grid:

- (a) of the string name, with **Name**
- (b) and any of the string attributes, with **Att Key**

then the attributes in the **Map Attributes** field are created as string attributes of the string.

Once a match has been found, no other searching is done in the grid.

#### Attributes >Vertex/Pit



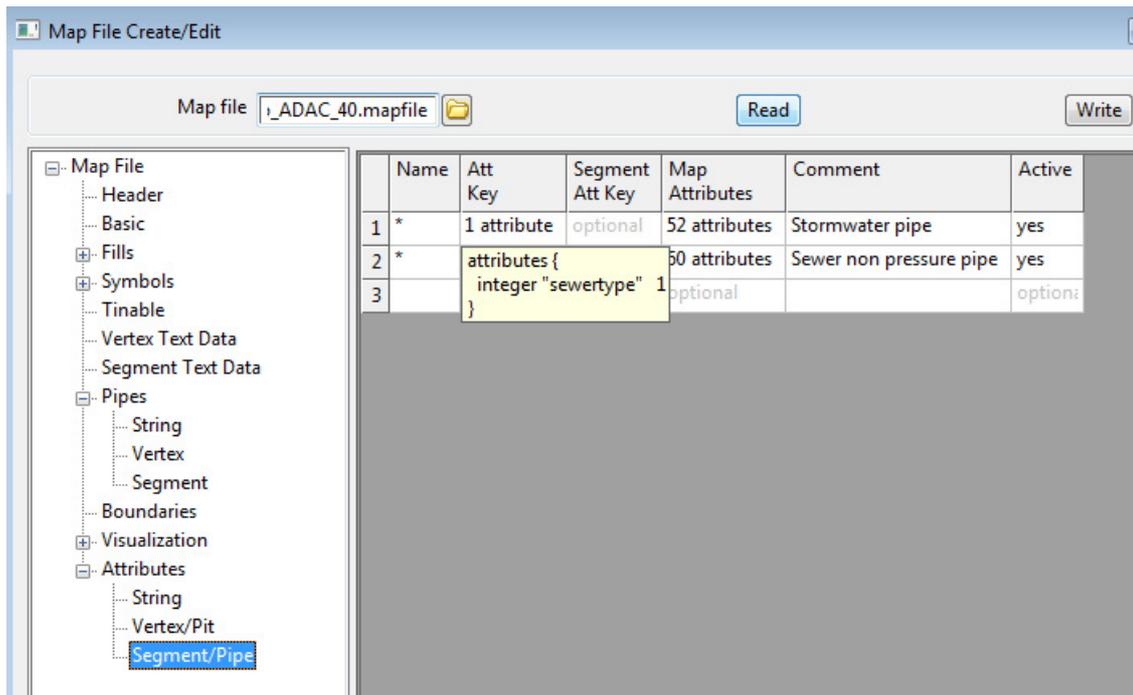
When the vertex of a string finds a match in the grid:

- (a) of the string name, with **Name**
- (b) and any of the string attributes, with **Att Key**
- (c) and any of the vertex attributes, with **Vertex Att Key**

then the attributes in the **Map Attributes** field are created as vertex attributes to the vertex of the string.

Once a match has been found, no other searching is done in the grid.

**Attributes >Segment/Pipe**



When the segment of a string finds a match in the grid:

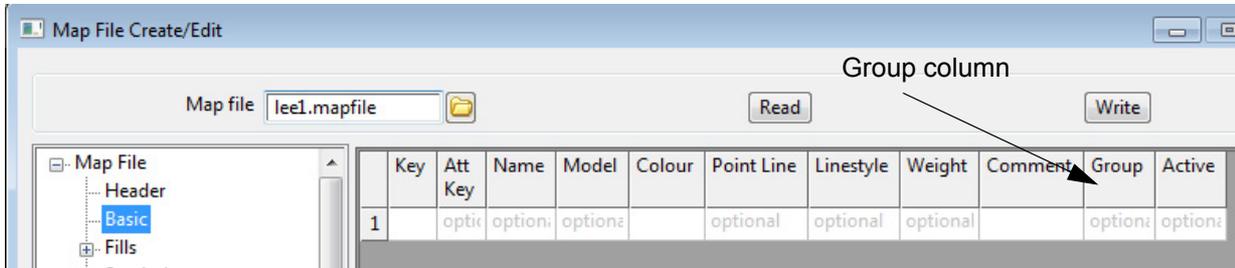
- (a) of the string name, with **Name**,
- (b) and any of the string attributes, with **Att Key**
- (c) and any of the segment attributes, with **Segment Att Key**,

the attributes in the **Map Attributes** field are created as segment attributes to the segment of the string.

Once a match has been found, no other searching is done in the grid.

### 4.3.3 Group Column in Grids

There is now an optional **Group** column in the grid for the **Basic** node in the **Map File** so that the individual rows of the grid can be given a group/subgroup structure. This has no use in the **Map File** but is used in the **Names.4d** file which has the same XML structure (see [4.4.1 Group Column in Names.4d](#)).



**IMPORTANT NOTE:**

Regardless of the group/subgroup structure, the search order for **finding a match** with the key in the **Basic** node of the **Map File** is still the order that the rows occur in the **Basic** grid.

Currently the Group Column will only be visible if you have the environment variable **USE\_TREE\_NAME\_BOX\_4D** set to 1.

## 4.4 Names.4d File

See

[4.4.1 Group Column in Names.4d](#)

[4.4.2 Active Column in Grids in Names.4d](#)

[4.4.3 Pipe Node in Names.4d](#)

[4.4.4 Attributes Node in Names.4d](#)

### 4.4.1 Group Column in Names.4d

There is now an optional **Group** column in the grid for the **Basic** node in the **Names.4d** file so that the individual rows of the grid can be given a group/subgroup structure.

If the Group column for a row is not blank, then the information is used as a group/subgroup structure in any names.4d pop up.

The group/subgroup structure is written as the top group name first, followed by the first level subgroup name, the second level subgroup name, etc., with each of the names separated by a forward slash /.

group\_name/first\_level\_subgroup\_name/second\_level\_subgroup\_name etc.

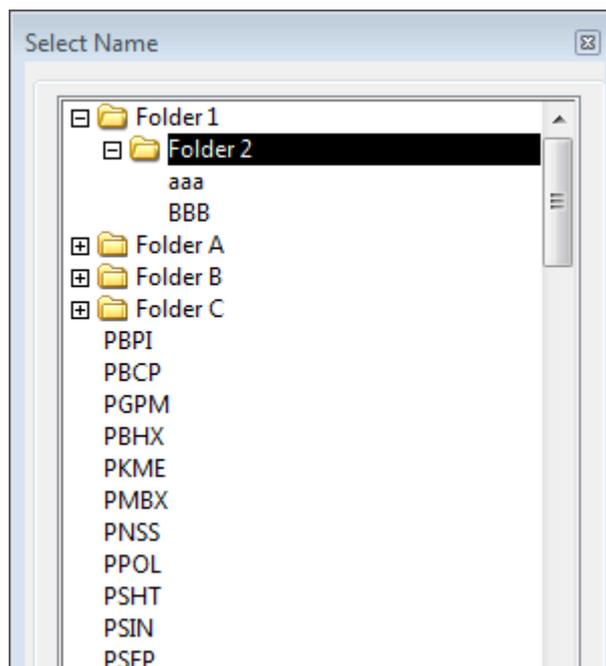
For example, a top level group *Folder1* with the first level subgroup *Folder2* is written as

Folder1/Folder2

**IMPORTANT NOTE:**

Regardless of the group/subgroup structure, the search order for **finding a match** with the key in the **Basic** node of the **Names.4d** file is still the order that the rows occur in the **Basic** grid.

In a **Names.4d** pop, the above names.,4d file will look like



If you **don't** want the **Group Column** to be used then you have to set the environment variable **USE\_TREE\_NAME\_BOX\_4D** to **0**. By default it has the value **1**.



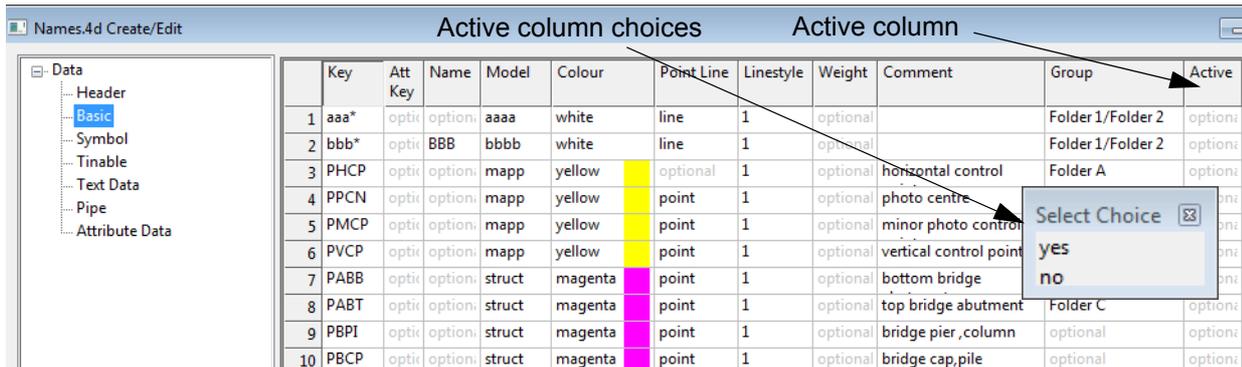
## 4.4.2 Active Column in Grids in Names.4d

There is now an optional **Active** column in each grid in the **Names.4d File** so that the individual rows of the grid can be turned off and so not used in the Names.4d File. This replaces having to delete the row.

If the value in the **Active** column for a row column is left **blank**, then the row is used.

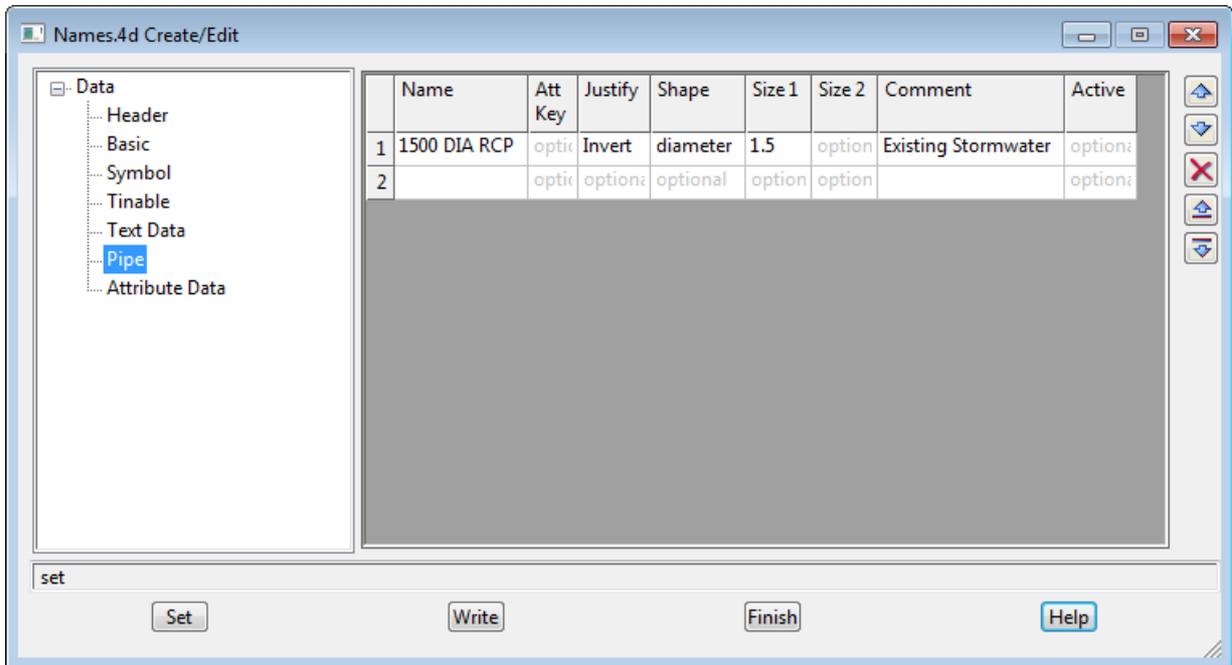
If the value in the **Active** column for a row column is **yes**, then the row is used.

If the value in the **Active** column for a row column is **no**, then the row is **not** used.



### 4.4.3 Pipe Node in Names.4d

There is a new **Pipe** node in the **Names.4d File** which is used to give a string a constant round pipe or a constant culvert pipe.



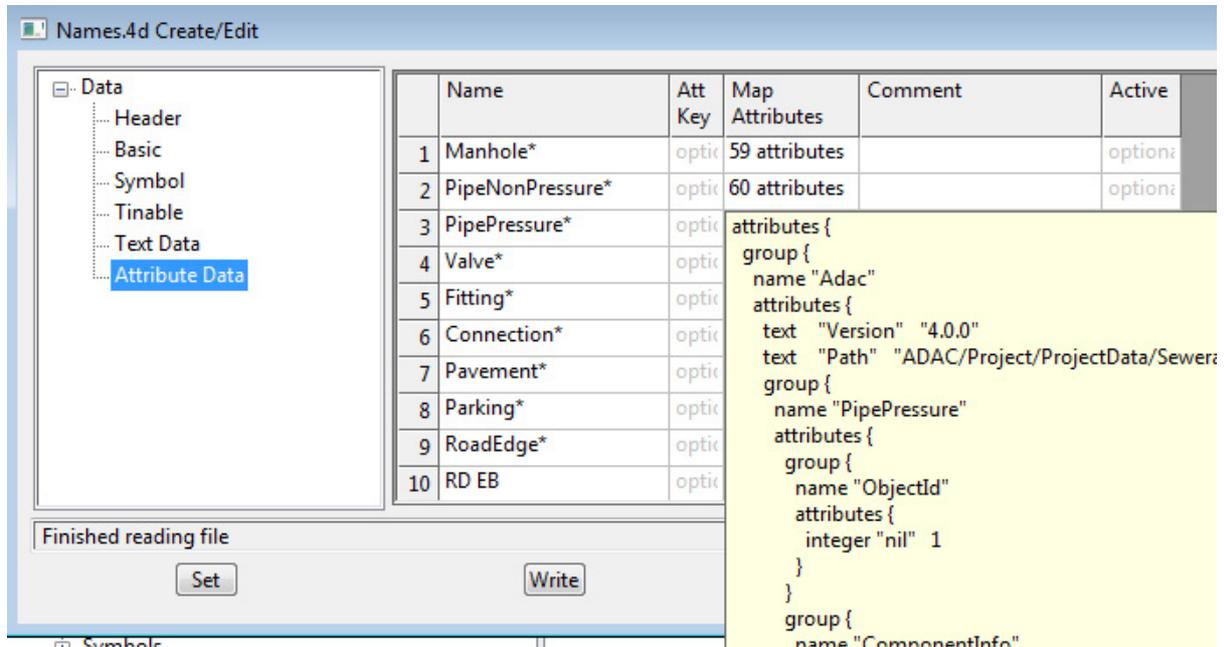
If a string name is entered in the **Name** field of the CAD ControlBar (by selecting from the Names pop up or by typing in a name and then pressing <Enter>) and there is a match in the **Pipe** section of **Names.4d**, then the values for that row in the **Justify**, **Shape**, **Size 1** and **Size 2** columns of **Names.4d** are placed in the **Pipe ControlBar** (see [12.14 Pipe ControlBar](#)).



Note that there does not need to be a match in the **Basic** section of **Names.4d**, just a match in the **Pipe** section of **Names.4d**. BUT if it is not in the **Basic** section of the **Names.4d** file then it won't appear in the **Names** pop up and the name will have to be typed into the **Name** field by hand followed by <Enter>.

## 4.4.4 Attributes Node in Names.4d

There is a new **Attributes Data** node in the **Names.4d File** which is used to give attributes to strings as string attributes.



If a string name is selected in the **Name** section of the CAD ControlBar (by selecting from the Names pop up or by typing in a name and then pressing <Enter>) **and** there is a match in the **Attributes Data** section of **Names.4d**, then attributes in the **Map Attributes** column of **Names.4d** are placed in the **Attributes ControlBar** (see [12.15 Attributes ControlBar](#)).



Note that there does not need to be a match in the **Basic** section of **Names.4d**, just a match in the **Attributes Data** section of **Names.4d**, BUT if it is not in the **Basic** section of the **Names.4d** file then it won't appear in the **Names** pop up and the name will have to be typed into the **Name** field by hand followed by <Enter>.

# 5. Drag and Drop

Some files can now be dragged and dropped onto **12d Model** and panels automatically open for them, and some can be dragged and dropped as attachments to Outlook 2002 and above.

**.project** files can be dragged and dropped to open a new project and files can be dragged and dropped onto a file box and the full path name to the file will be automatically entered into the file box.

See

[5.1 File Box](#)

[5.2 .12daz, 12da Files](#)

[5.3 .dat Files](#)

[5.4 .chain,.rcn Files](#)

[5.5 .mtf Files](#)

[5.6 .mapfile Files](#)

[5.7 .slx Files](#)

[5.8 .12dpsf Files](#)

[5.9 .4do Files](#)

[5.10 .project Files](#)

[5.11 .dwg,.dxf Files](#)

## 5.1 File Box

If you drag a file from *Windows Explorer* onto a **File Box** on any panel, the full file path name of the file is entered into the **File Box**.

## 5.2 .12daz, 12da Files

If you drag a **.12daz** or **.12da** file onto an open **12d Model** project, it opens up the **Read 12d Solutions Archive Data** panel with the full path name of the **.12daz** or **.12da** file entered into the **File to read** box.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

It will also work for files that are email attachments in *Outlook 2002 and above*.

If you are dragging and dropping more than one **12daz** or **12da** file at a time, then the one **Read 12d Solutions Archive Data** panel will be opened and all the dropped files listed in the **Many Files** grid.

## 5.3 .dat Files

If you drag a **.dat** file onto an open **12d Model** project, it opens up the **Read x y z s Data** panel with the full path name of the **.dat** file entered into the **File to read** box.

This will work for files from folders, Skype and most other places that allow drag and drop.

It will also work for files that are email attachments in *Outlook 2002 and above*.

If you are dragging and dropping more than one **.dat** file at a time, then the one **Read x y z s Data** panel will be opened and all the dropped files listed in the **Many Files** grid.

## 5.4 .chain,.rcn Files

If you drag a **.chain** or **.rcn** file onto an open **12d Model** project, it opens up the **Create/Edit Chain** panel with the full path name of the **.chain** or **.rcn** file entered into the **Chain file** box and the chain file is automatically read in.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

If you are dragging and dropping more than one **chain** file at a time, then a separate **Create/Edit Chain** panel is opened for each chain file.

## 5.5 .mtf Files

If you drag a **.mtf** file onto an open **12d Model** project, it opens up the mtf in the **MTF Edit** panel.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

If you are dragging and dropping more than one **mtf** file at a time, then a separate **MTF Edit** panel is opened for each mtf file.

## 5.6 .mapfile Files

If you drag a **.mapfile** file onto an open **12d Model** project, it opens up the **Map File Create/Edit** panel with the full path name of the **.mapfile** file entered into the **Map file** box and the Map File is automatically read in.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

If you are dragging and dropping more than one **Map File** at a time, then a separate **Map File Create/Edit** panel is opened for each Map File.

## 5.7 .slx Files

If you drag an **.slx** file onto an open **12d Model** project, it opens up all the panels as specified in the **SLX** file.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

If you are dragging and dropping more than one **SLX** at a time, then all the panel in all the **SLX** files are opened.

## 5.8 .12dpsf Files

If you drag a **.12dpsf** file onto an open **12d Model** project, it opens up the **Create/Edit Plot Sheet** panel and reads the dropped **12dpsf** file into the panel.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

If you are dragging and dropping more than one **12dpsf** at a time, then a separate **Create/Edit Plot Sheet** panel is opened and the file read in and processed for each **12dpsf** file.

## 5.9 .4do Files

If you drag a **.4do** file onto an open **12d Model** project, it runs the macro.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

If you are dragging and dropping more than one **.4do** file at a time, then each macro will be started.

## 5.10 .project Files

If you drag a **.project** file (the **12d Model** project) onto the **12d Model 32** icon or **12d Model 64** icon, the dragged **.project** is opened in **12d Model**.

If you drag a **.project** file (the **12d Model** project) onto an open **12d Model** project, the current open **12d Model** project is closed (it will ask whether you want to save) and the dragged **.project** is opened in **12d Model**.

## 5.11 .dwg,.dxf Files

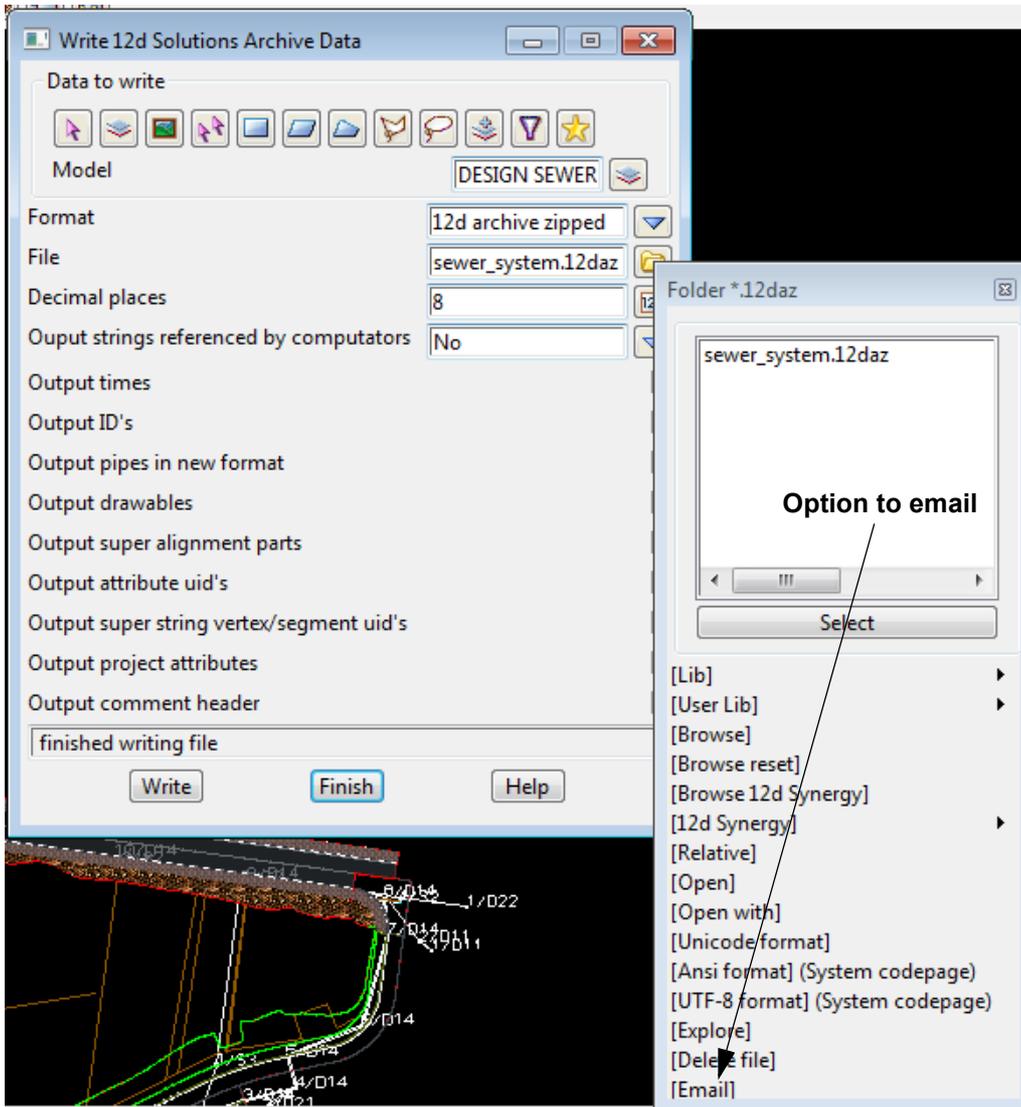
If you drag a **.dwg** or **.dxf** file onto an open **12d Model** project, it opens up the read **Read DWG/DXF Data** panel with the full path name of the **.dwg/.dxf** file entered into the **File** box.

This will work for files from folders, 12d Synergy, Skype and most other places that allow drag and drop.

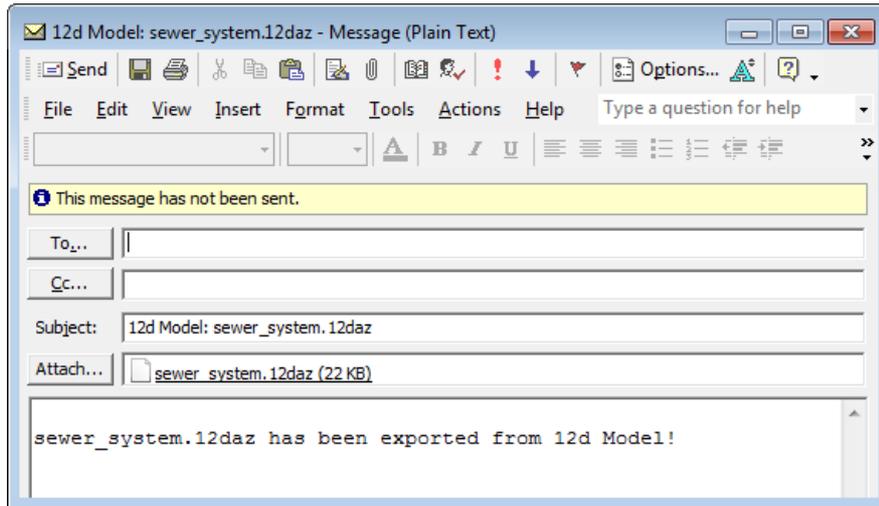
If you are dragging and dropping more than one **DWG/DXF File** at a time, then a separate **Read DWG/DXF Data** panel is opened for each DWG/DXF file.

# 6. Emailing from File Boxes

When a **File box** has a file name in it and the file exists (this usually means the option has to be run first), clicking on the **File** icon at the end of the field will bring up the **Folder** pop-up and at the bottom there is now an **[Email]** choice.



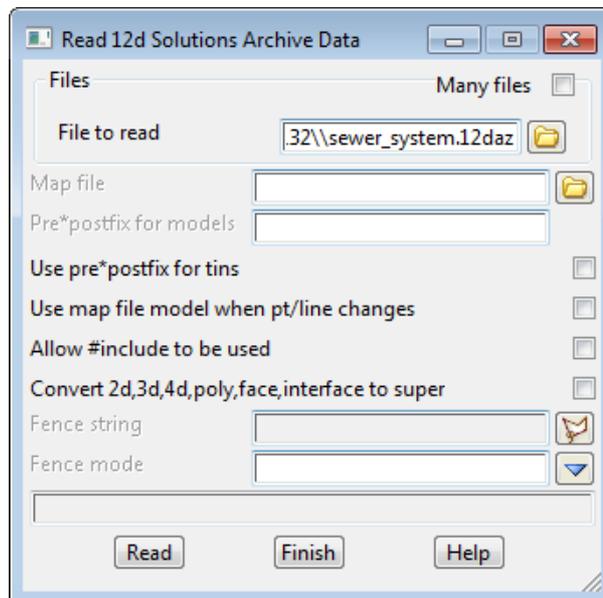
If you have *Outlook 2002* and above (or another **MAPI** registered application), clicking on **[Email]** will create an email with that file as an attachment ready for you to give the email address and any extra information you would like to put in the email.



This will work with Outlook 2002 and above, and should also work with other **MAPI** registered applications.

In the above example, the 12d Archive File was written out as a zipped 12da file (a 12daz file). Zipping the 12da file can result in files a twentieth of the size of the original 12da file (*i.e.* 5% of the original), which makes it much better for emailing around.

Also, when you receive a 12daz or a 12da file as an attachment to an email, you can drag and drop it onto an open 12d project and it will automatically open the **Read 12d Solutions Archive Data** panel with the dropped file name already filled in as the **File to read**.



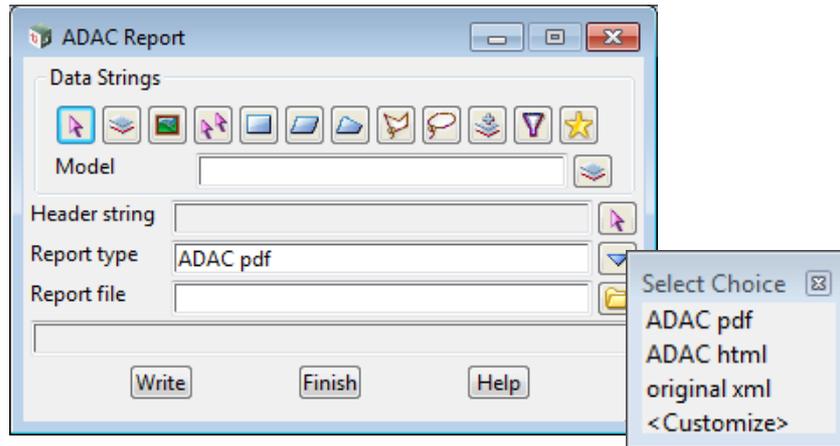
### Important Note

The file must already exist **before** the **[Email]** on the pop up can be used so usually the Panel option must be run first to create the file, and then go back and click on the **File** icon to bring up the pop up for the file to be emailed.

# 7. Setting Up XML Reports

Many options now have **XML reports** and allow users to create their own customised reports.

And once you have a customised report for a panel, you then need to add it to the list of reports given in the **Report type** field for that panel.

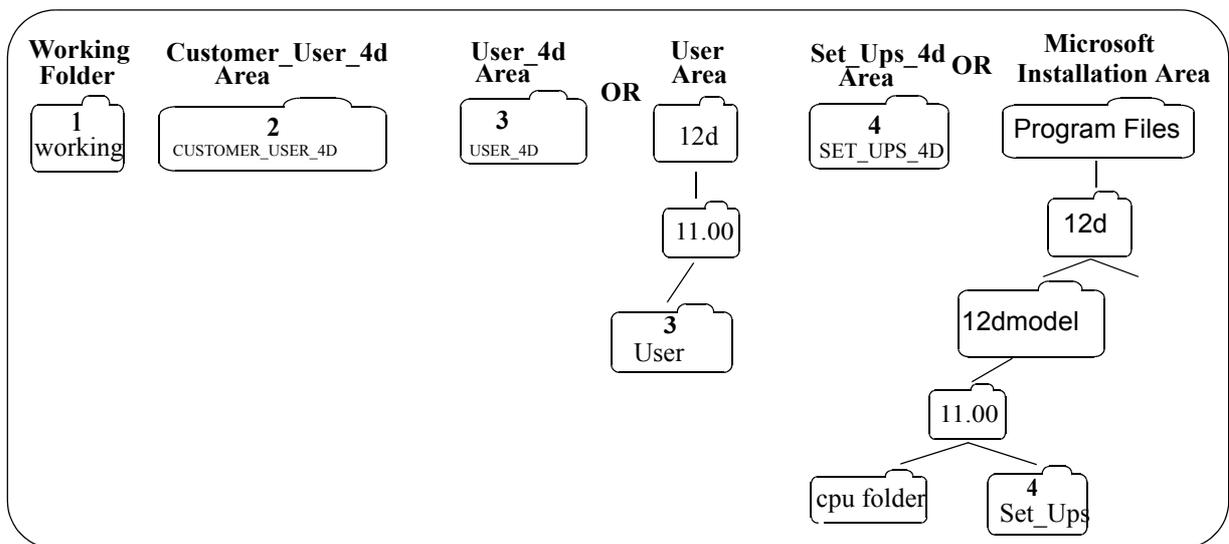


In **12d Model 11**, there is a new XML file called **report\_templates.xml** that defines which reports, 12d supplied or user customised, are available for a panel.

For each of the reports for a panel, the **report\_templates.xml** file contains

- (a) the name in the **Report type** pop-up list
- (b) the xslt that generates the report
- (c) the file ending for the report file
- (d) the number of decimal places in the report
- (e) a comment about the report

The **report\_templates.xml** are all **Set\_Ups** files like *colours.4d files*, and folders are searched in a specific order to find **report\_templates.xml** files.



**HOWEVER**, unlike *colours.4d*, it doesn't matter if the **report\_templates.xml** file is found in one folder, the search **continues** and **all** the **folders** that **contain the file** are found.

And, again unlike *colours.4d*, the reports used are the **merger** of **all** the **found files**.

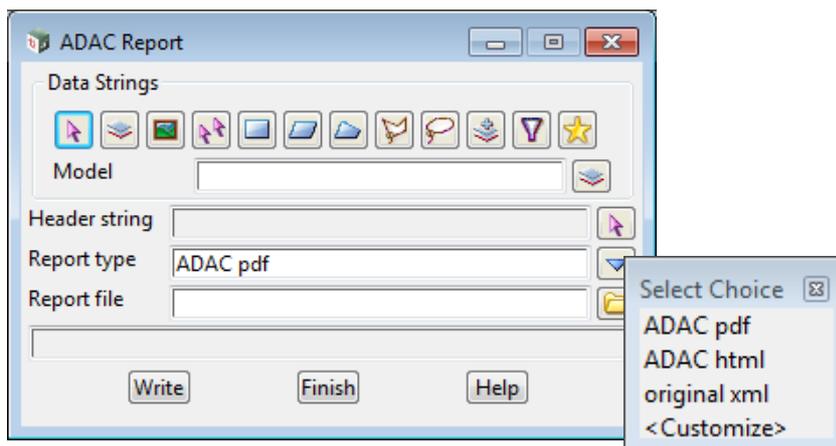
The search order is still important because if the **same name for a report for a panel** exists in more than one file in the found folders, the **first** occurrence in the files in the search order is the one that **is used**.

For example, if you had a report called **ADAC pdf** in the file *report\_templates.xml* in your working folder, and one called **ADAC pdf** in the file *report\_templates.xml* in Programs files\12d\12dmodle\11.0\set\_ups, then the **ADAC pdf** in your **working folder** is the one that will be used and seen in the **Report type** pop-ups in panel **ADAC Report**.

To create and/or edit the **Report types** for a panel, see [7.1 Create/Edit Report Types](#).

## 7.1 Create/Edit Report Types

To create and/or edit the **Report types** for a panel, click on the choice icon for the **Report types** field and select **<customise>**



The **Edit Report\_templates.xml** panel is then brought up and it shows the standard areas for looking for the **report\_templates.xml** files - Working folder, Customer (User), User, Set\_Ups and Other - and displays whether the appropriate xml file:

- (a) exists and is available for editing (**Edit**)

If the xml file exists in the folder and you have access to edit it, then the folder name is written in the panel and **[Edit]** is written on the right hand side of the line.

- (b) does not exist but can be created (**Create**) by the editor

If the xml file does not exist in the folder and you have access to create it, then the folder name is written in the panel and **[Create]** is written on the right hand side of the line.

In this case a new file is created and written out to the given folder.

- (c) is in a folder that has no access for editing and so can only be viewed (**View**)

If there is no access to the folder to create/edit the file, then the folder name is written in the panel and **[View]** is written on the right hand side of the line. Although the styles can't be edited, they can be **copied**.

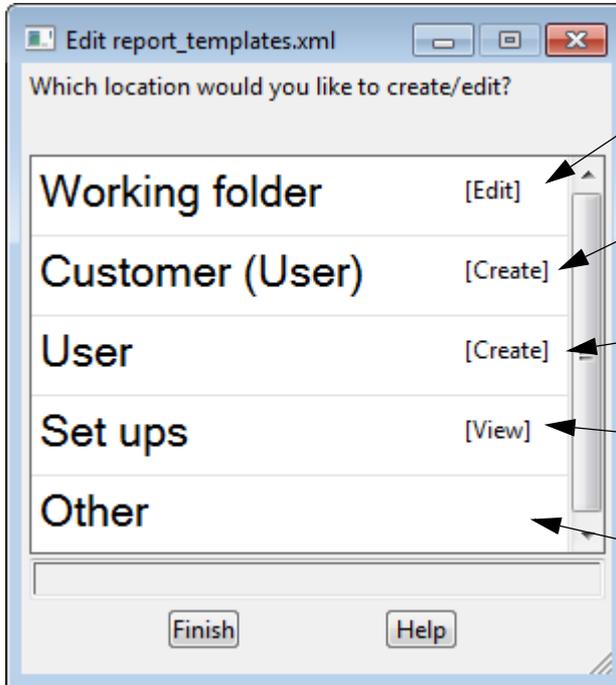
This usually applies to the **Set Ups** folder which normally requires Admin privileges for it to be written to.

(d) is in **Customer (User)**.

This only appears if a *Customer User* area has been defined.

(e) is defined separately by the user and then **Other** is displayed.

In this case, the full path name of the XML file is given by the user.



**[Edit]** - there is an xml file and it can be edited

**[Create]** - there is no xml file but one can be created. **Customer (User)** only appears if you have defined a Customer User area

**[Create]** - there is no xml file but one can be created.

**[View]** - no xml file can be created because there is no access, just viewed.

nothing means that the user has not defined any additional files to use

See

[7.1.1 \[Create\] for Report Types](#)

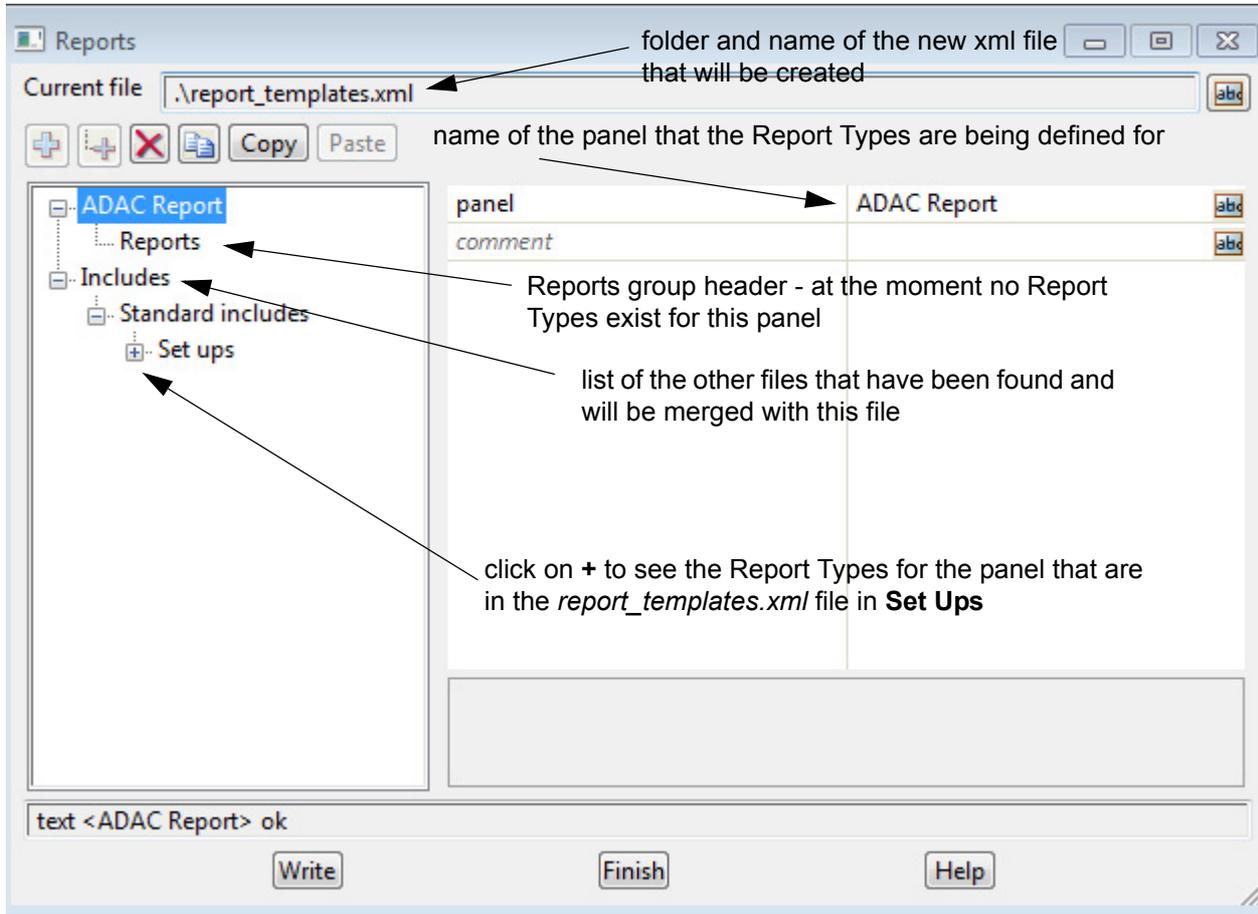
[7.1.2 \[Edit\] for Report Types](#)

## 7.1.1 [Create] for Report Types

When **[Create]** is shown, a new xml file can be created in that folder.

Click on the line with **[Create]** and the **Editor** for a **report\_templates.xml** starts up with a **Reports** group header but reports in it.

All the other **report\_templates.xml** files that are found are listed in the **Includes** section displayed at the bottom of the tree.



The included files can not be edited whilst create/editing this file BUT you can copy items from them and paste them into items in this file.

New items for the **Report type** pop up can now be created (see [7.1.2.1 Creating New Report Types](#)) and a new XML created by clicking on the **Write** button.

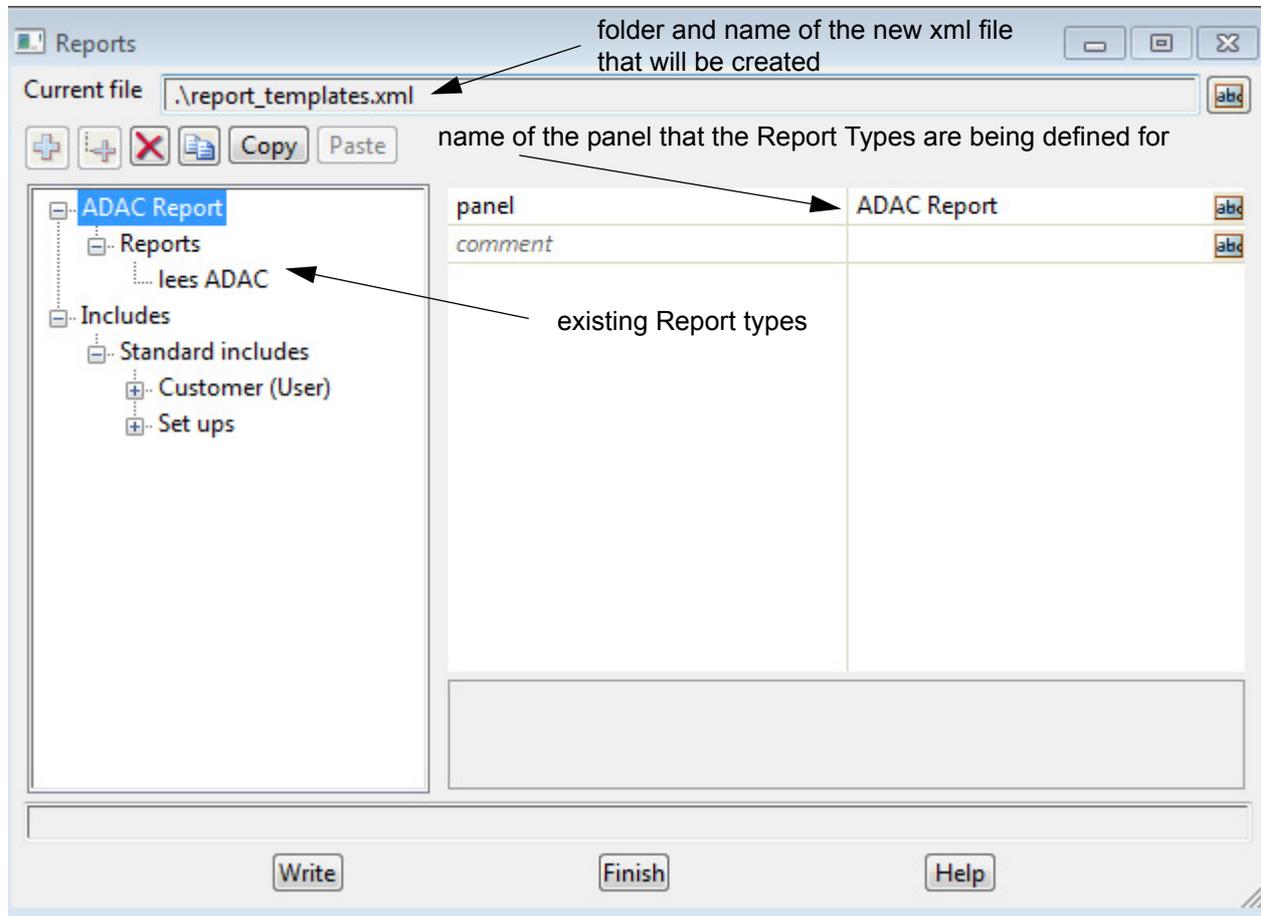
Creating and editing of *report\_templates.xml* files is the same as for **[Edit]** and so is documented in that section. See [7.1.2 \[Edit\] for Report Types](#).

**Write** - writes out the information in the panel to the files name given in **Current file**.

## 7.1.2 [Edit] for Report Types

When **[Edit]** is shown, an xml file already exists and can be edited.

Click on the line with **[Edit]** and the **Editor** for the *report\_templates.xml* starts up and displays from the file, all the existing **Report types** for the panel.



After completing any editing, the XML is saved by clicking on the **Write** button.

As in the **[Create]** case, all the other appropriate files that are found are listed in the Includes section are displayed at the bottom of the tree (see [7.1.1 \[Create\] for Report Types](#)).

The included files can not be edited whilst create/editing this file BUT you can copy items from them and paste them into items in this file.

For editing in the panel, see

[7.1.2.1 Creating New Report Types](#)

[7.1.2.2 Copy and Paste](#)

[7.1.2.3 Deleting a Report Type](#)

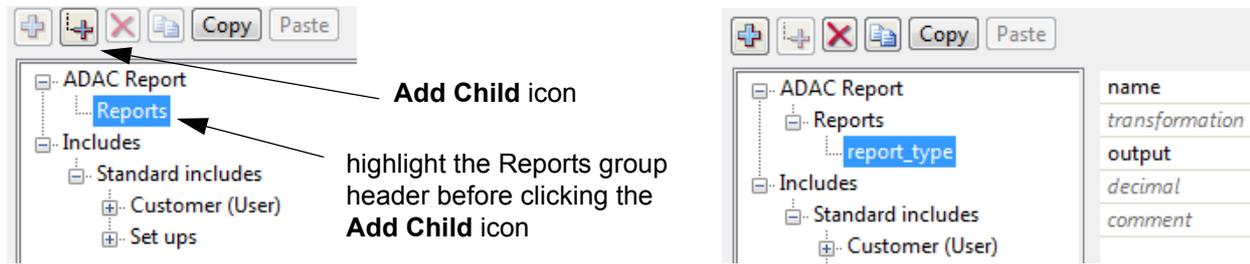
[7.1.2.4 Duplicating a Report Type](#)

[7.1.2.5 Editing a Report Type](#)

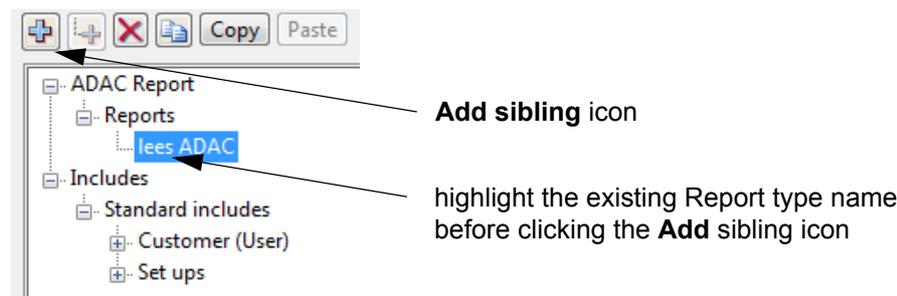
### 7.1.2.1 Creating New Report Types

To create a **new Report type** from scratch when there are **no Report types**, highlight the **Reports group header** and click on the **Add Child** icon.

**Note** - the **Add Child** icon adds a node as a subnode of the highlighted node.

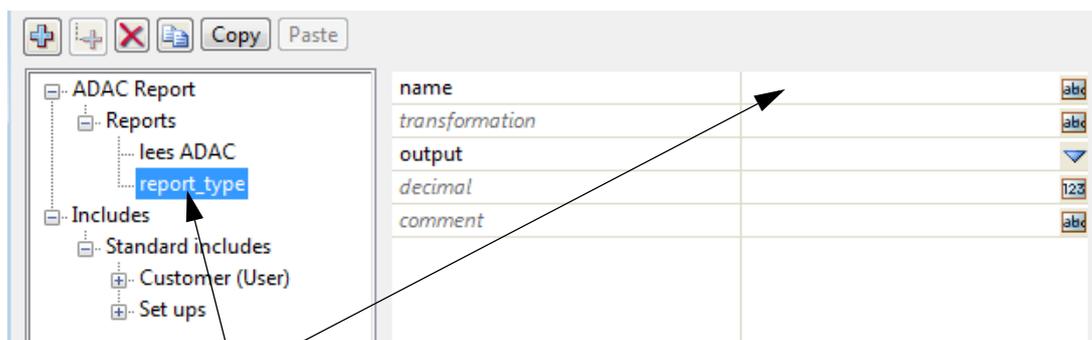


To create a **new Report type** from scratch when **some Report types already exist**, you can highlight the **Report type name** and click on the **Add** (Add Sibling) icon.



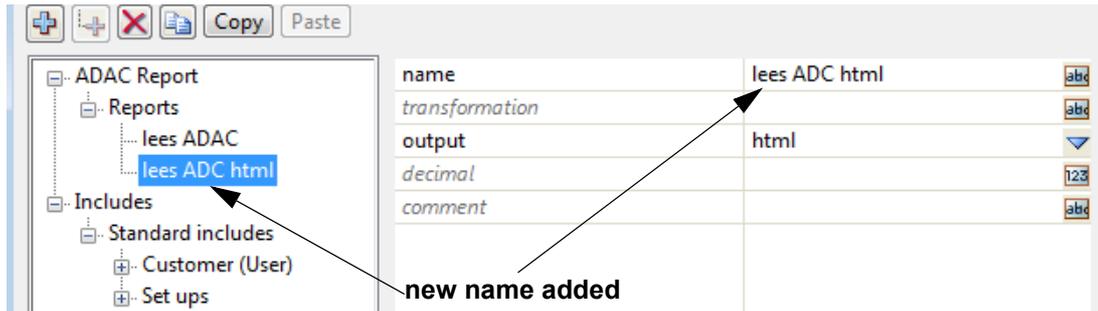
The **Add** icon adds a new item at **the same level** (a sibling) as the highlighted item and that is why a Report type must already exist so that one can be selected.

In both cases a new *Report type* is created with no name and so the general name **report\_type** is written in the *Reports* list until the *Report type* is given a name.



**new Report type added** - no name as yet

Once the new *Report type* is given a name then that name will appear in the tree.



Once the new *Report type* is created then it can be edited by going into each field of the *Report type* and making changes.

**Other Methods for Creating New Report Types**

**Duplicate** can be used if there are already existing *Report types* for the panel in the file ([7.1.2.4 Duplicating a Report Type](#)).

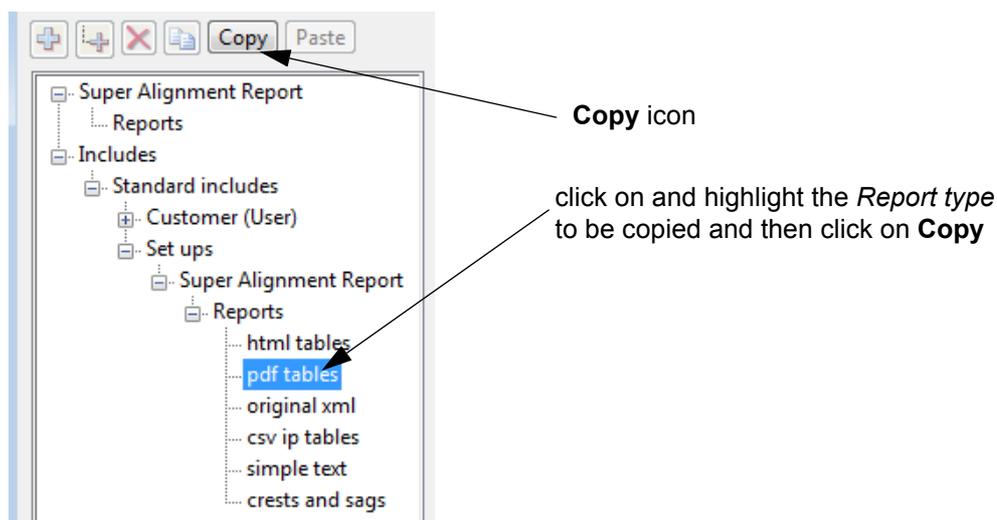
**Copy** and **Paste** is particularly useful when there are **no** *Report types* for the panel in the file but there are *Report types* for the panel in other files listed in the **Includes** node. Although these *Report types* can't be edited, they can be copied ([7.1.2.2 Copy and Paste](#)).

**7.1.2.2 Copy and Paste**

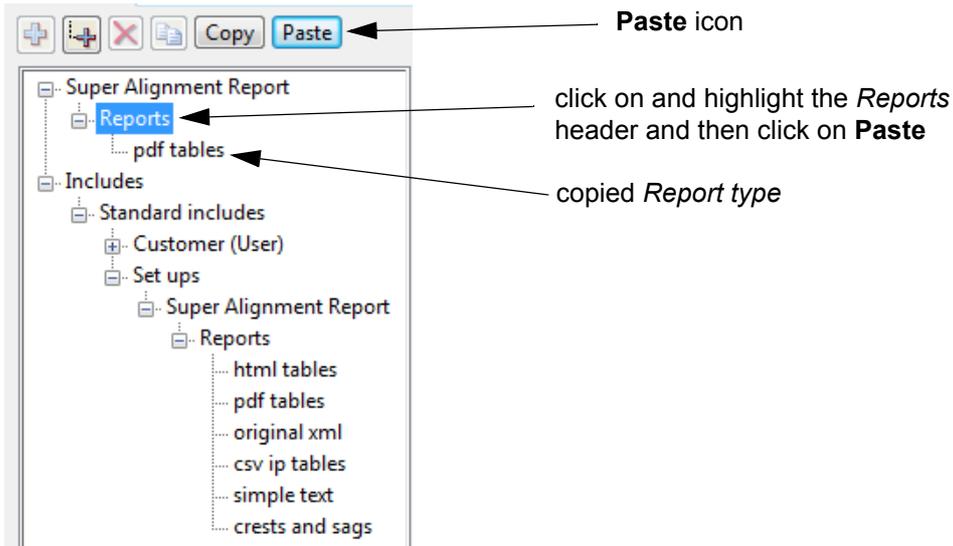
**Copy** and **Paste** can be used to create a copy of an existing *Report type*, and then the copied *Report type* edited.

Although there may be no *Report types* in the folder you are creating *Report types* for, there will usually be some *Report types* in the XML file in **Set Ups** that can be copied.

In the tree in the Editor, click on the **+** in front of **Set Ups** to expand the tree and then click on the item in the tree that you want to copy. Then click on the **Copy** icon.



Next click on and highlight the *Reports* header and click on **Paste**. The *Report type* will then appear under the *Reports* header with the same name as the original *Report type*. The copied *Report type* can then be edited.



**NOTE:** If the name of the copied *Report type* is **not changed** then this *Report type* will be used instead of the one in **Set Ups**.

### 7.1.2.3 Deleting a Report Type

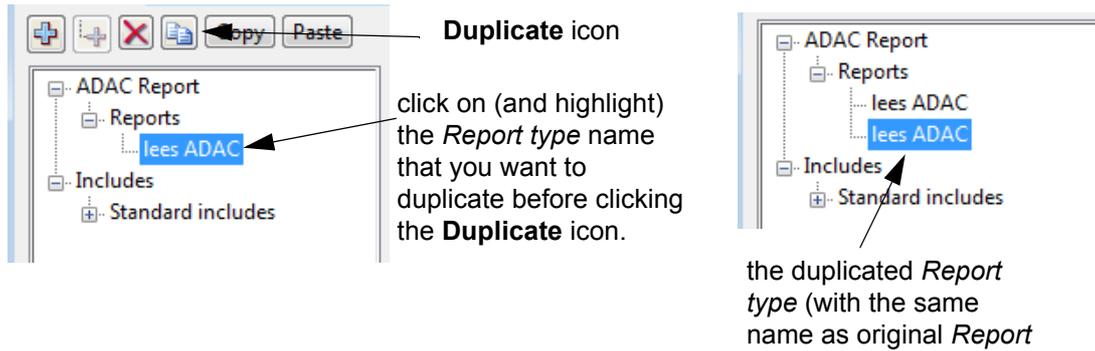
To **delete** a *Report type*, simply click on and highlight the *Report type name*, and then click on the **Delete** icon.



### 7.1.2.4 Duplicating a Report Type

To **duplicate** a *Report type*, simply click on and highlight the *Report type name*, and then click on the **Duplicate** icon.

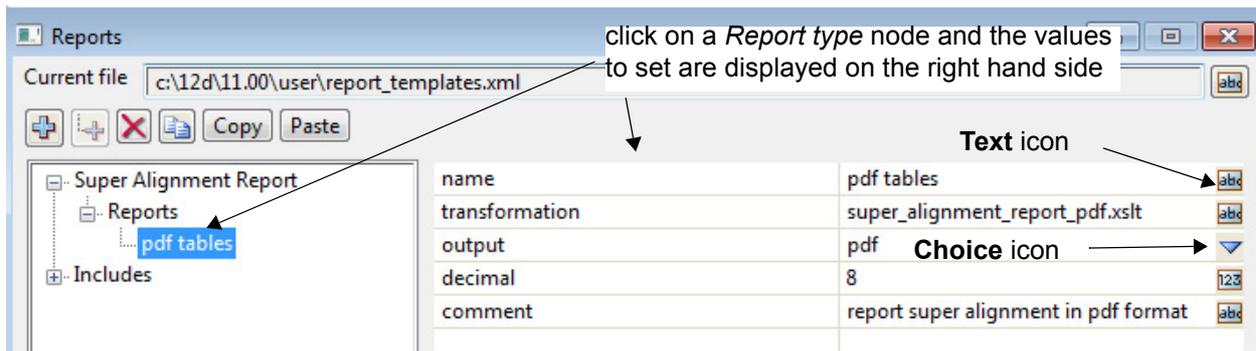
A copy of the *Report type*, with exactly the same name, is created and added to the end of the list of *Report type*.



The name of the Report type needs to be changed, and any other edits done that are required.

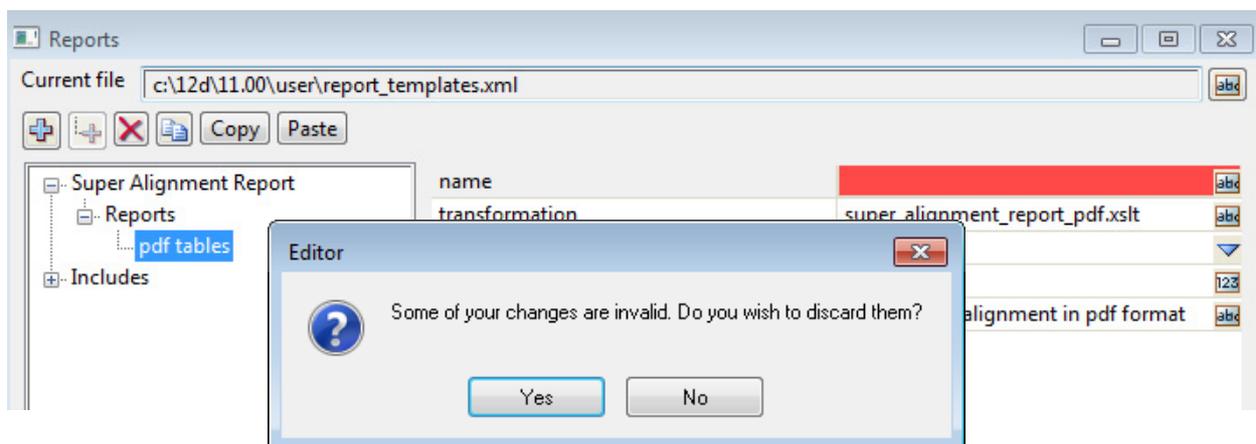
### 7.1.2.5 Editing a Report Type

To edit a Report type, click on the Report type name and all the values for that Report type will be displayed on the right hand side of the panel.



The panel fields on the right hand side of the panel have the same icons as in other 12d Model panels and the standard pop ups are all available. For example, text, number and choice icons.

When clicking to go to another Report type, all the fields currently on the right hand side are validated and if there are any problems, the first line with an error is coloured red and an Editor error box is displayed.



If Yes is selected, then all the invalid field values are discarded, the Editor error panel removed, and the data for the new selected node displayed on the right hand side of the panel.

If No is selected, then the right hand side of the panel does not change and can undergo further edits.

# 8. Sharing

See

[8.1 Local Caching of Share Data](#)

[8.2 Share Manager](#)

[8.3 Sharing Map Files](#)

[8.4 Localize Shares](#)

## 8.1 Local Caching of Share Data

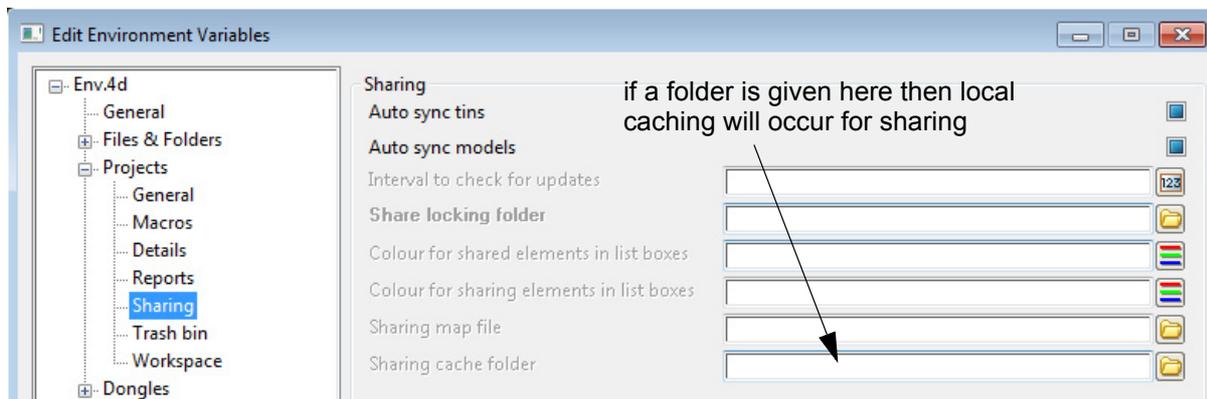
When models and tins are being shared **into** your project from across a network, the network speed can greatly affect performance. And this will happen every time the project is opened.

You can now have a **Shared Cache Folder** on your local drive, and it is used to keep local copies of the **shared in** models and tins.

It also means that any **shared in** models and tins that have not changed will already be in the local **Sharing Cache Folder** and not have to be accessed over the network again.

There is a new environment variable, SHARING\_CACHE\_4D, which gives the full pathname to the local folder to use as the **Sharing Cache Folder**.

In the **Edit Environment Variables** panel, it is the field **Sharing cache folder** in *Projects >Sharing*.



If **Sharing cache folder** is left blank then local caching does not occur.

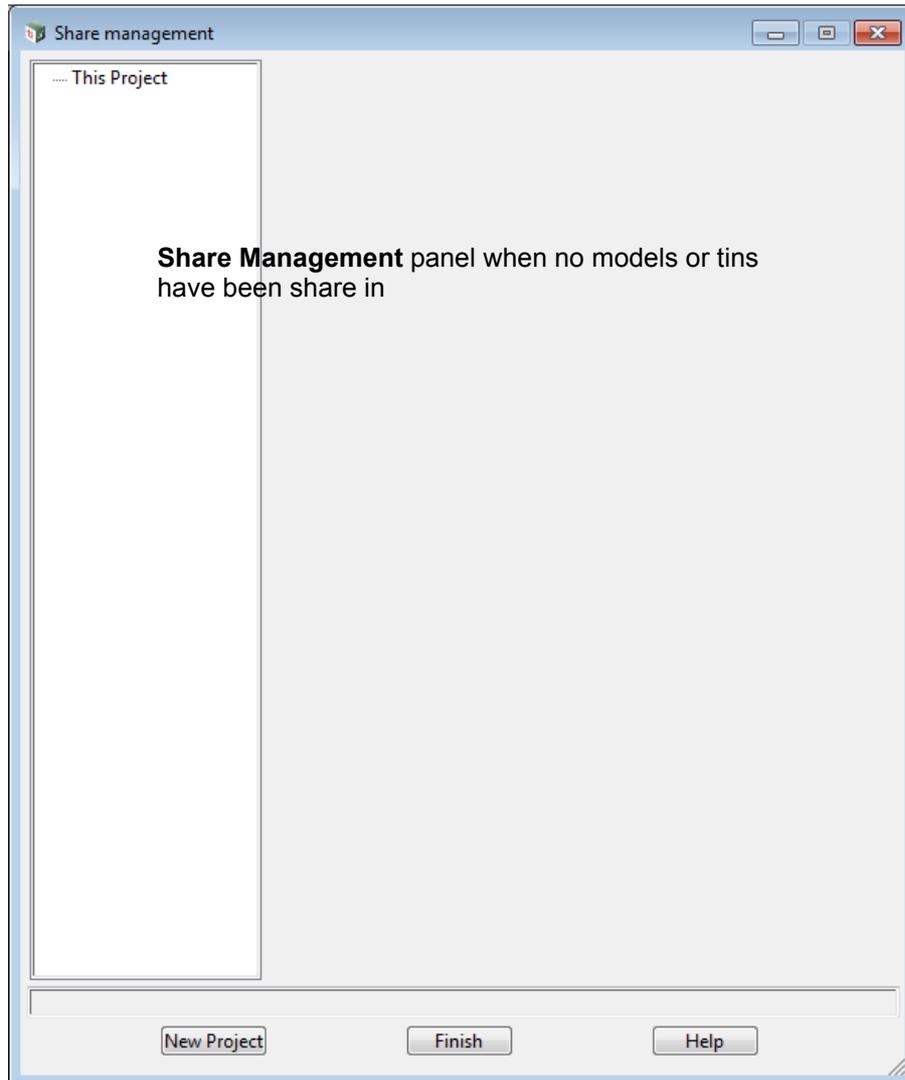
## 8.2 Share Manager

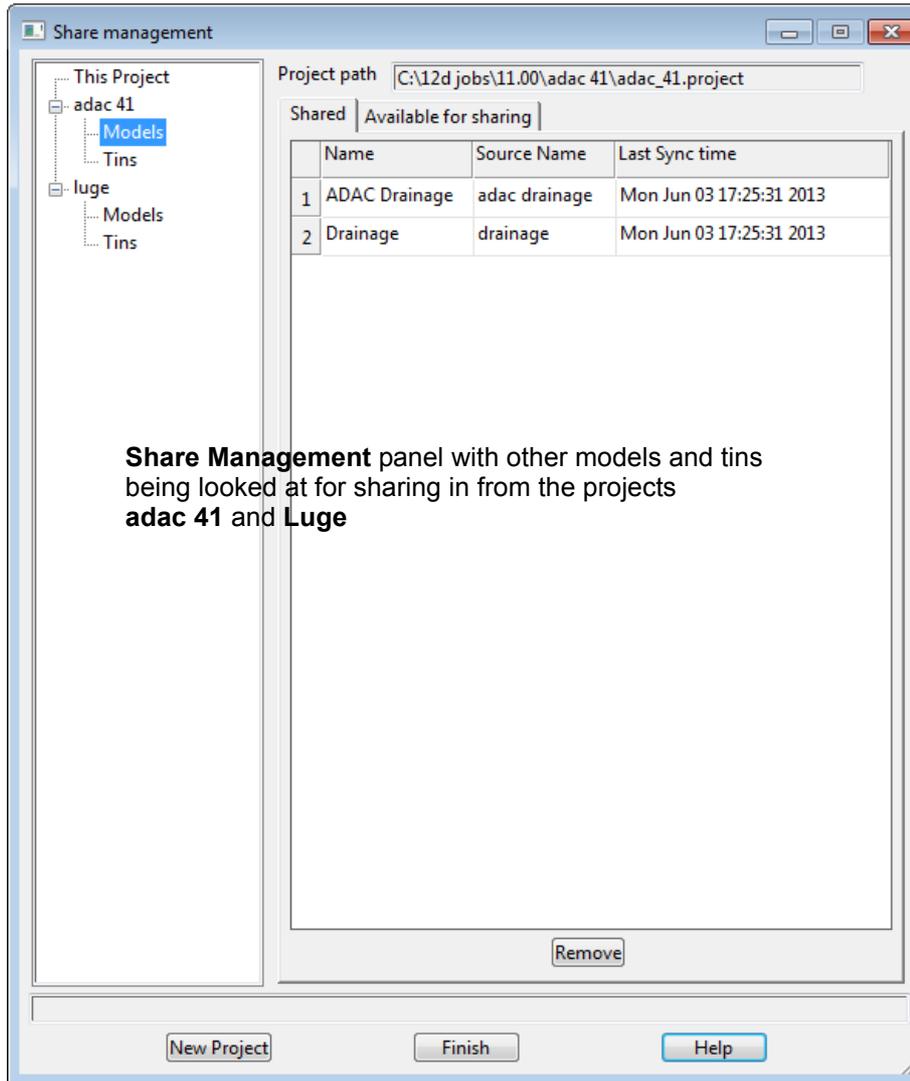
There is a new **Share Management** tool that combines the **Share**, **Add** and **Remove** options for Sharing models and tins.

Clicking on the option

Project =>Management =>Sharing =>Manage

brings up the **Share Management** panel and the panel will list under the **This Project** node, all the models and tins in the project that are available to share out, and as extra as nodes, all the projects that models and tins have been **shared in** from.





See

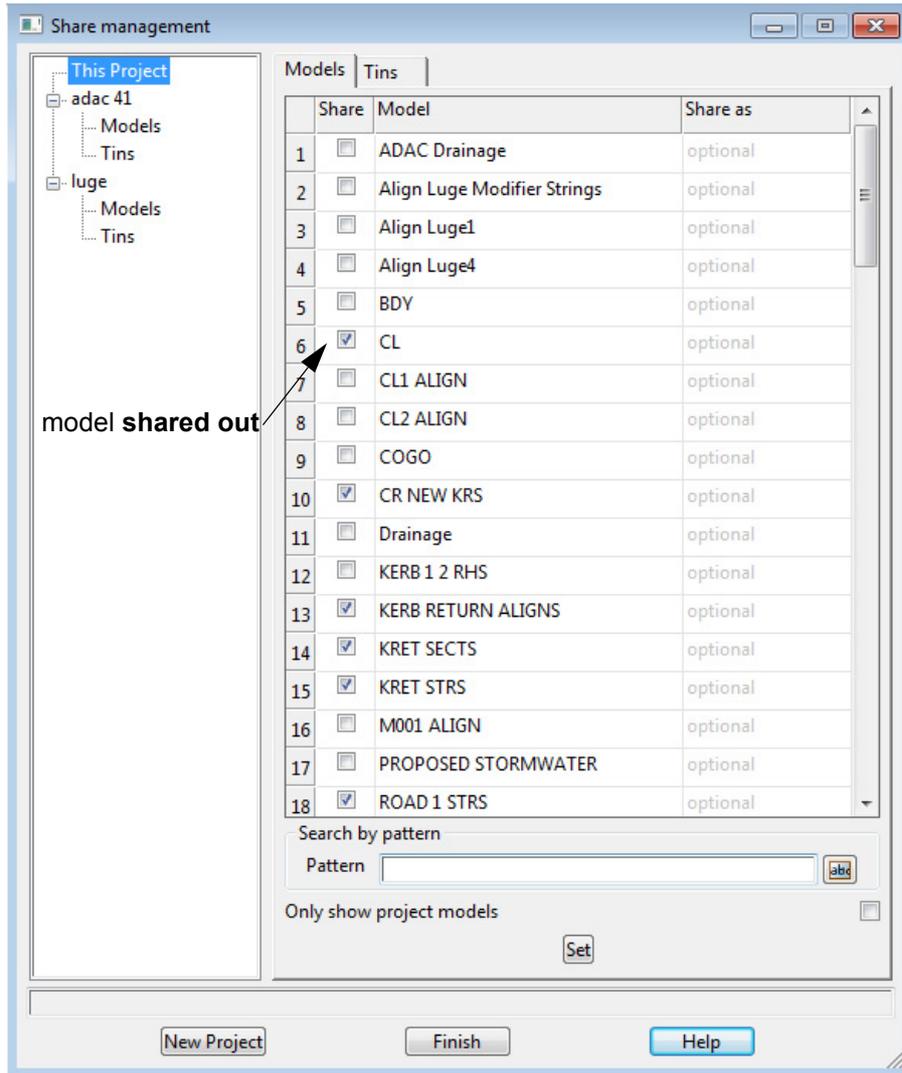
[8.2.1 Making Models or Tins Available for Sharing](#)

[8.2.2 Sharing in Models and Tins](#)

## 8.2.1 Making Models or Tins Available for Sharing

Clicking on **This Project** displays all the models in the project on the **Models** tab, and all the tins in the project under the **Tins** tab.

Models or tins that have been made available for sharing, that is are **shared out** so other projects can share in these models and tins from this project, have a tick beside them in the **Set** column.

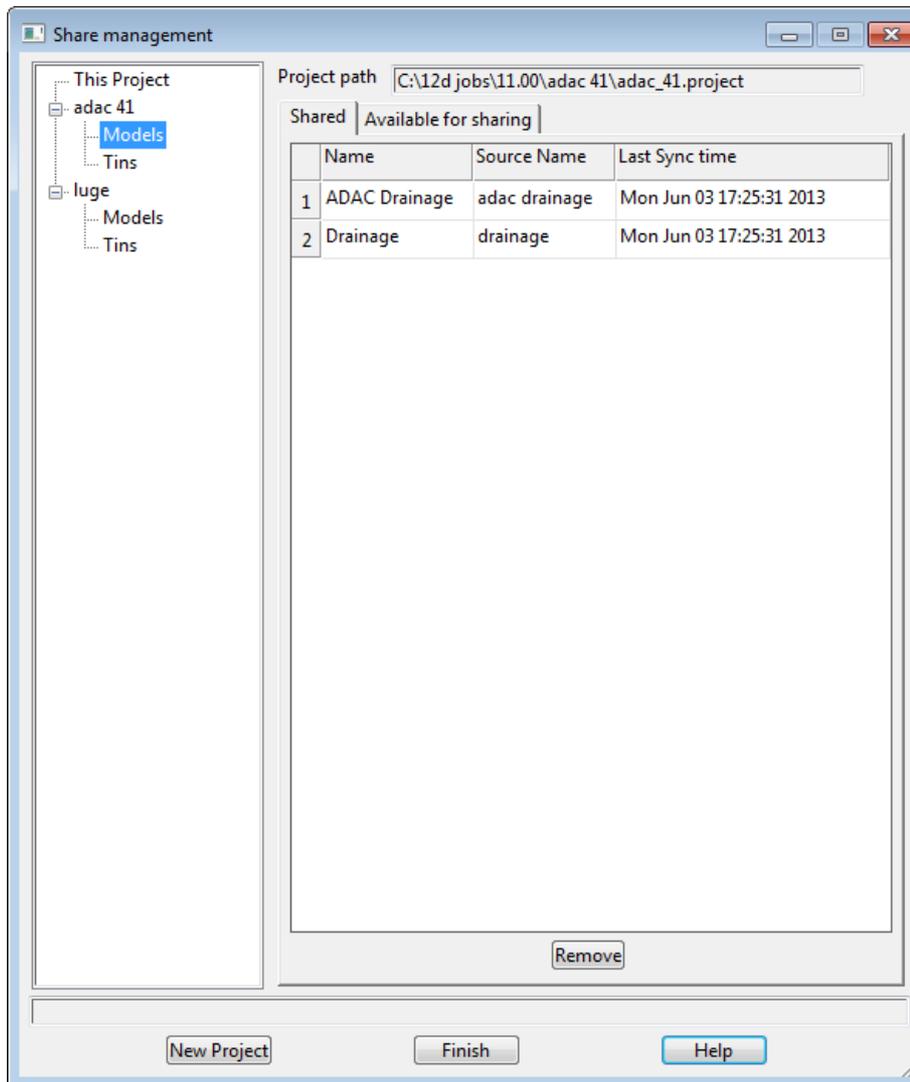


To share models or tins out, simply click the tin on in the **Share Out** column and then click on **Set**.

Similarly if there is no tick in the **Share** column then clicking on **Set** will make the models or tins no longer available to share out.

## 8.2.2 Sharing in Models and Tins

Under each project node, there is a **Models** and **Tins** node.



### Models

Clicking on the **Models** node lists all the models that have been shared in on the **Shared** tab, and all the models that are available to be shared in from that project on the **Available for Sharing** tab.

On the **Shared** tab, highlighting a model name and clicking on the **Remove** button will stop the model from being shared in.

On the **Available Sharing** tab, all models available for sharing in are shown and those that have already been shared in are coloured yellow. Models that are not shared in can be shared in by ticking in the **Share ?** column and then clicking the **Add** button.

### Tins

Clicking on the **Tins** node lists all the tins that have been shared in on the **Shared** tab, and all the tins that are available to be shared in from that project on the **Available for Sharing** tab.

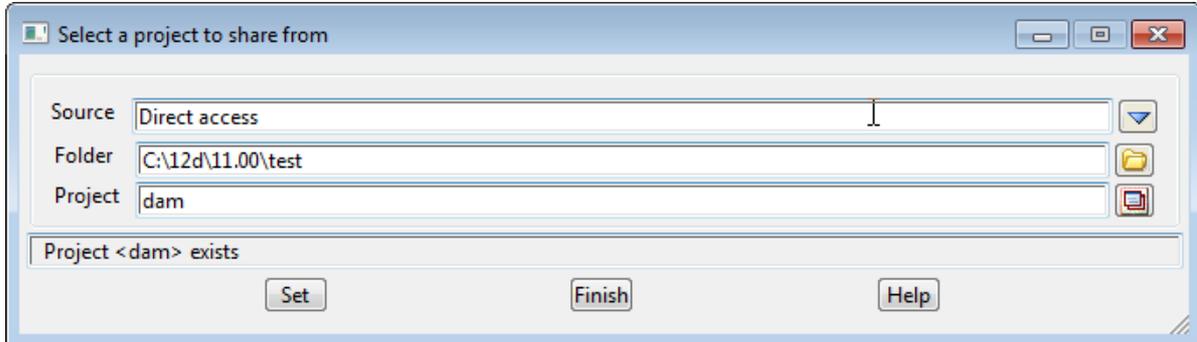
On the **Shared** tab, highlighting a tin name and clicking on the **Remove** button will stop the ting from being shared in.

On the **Available Sharing** tab, all tins available for sharing in are shown and those that have already been shared in are coloured yellow. Tins that are not shared in can be shared in by

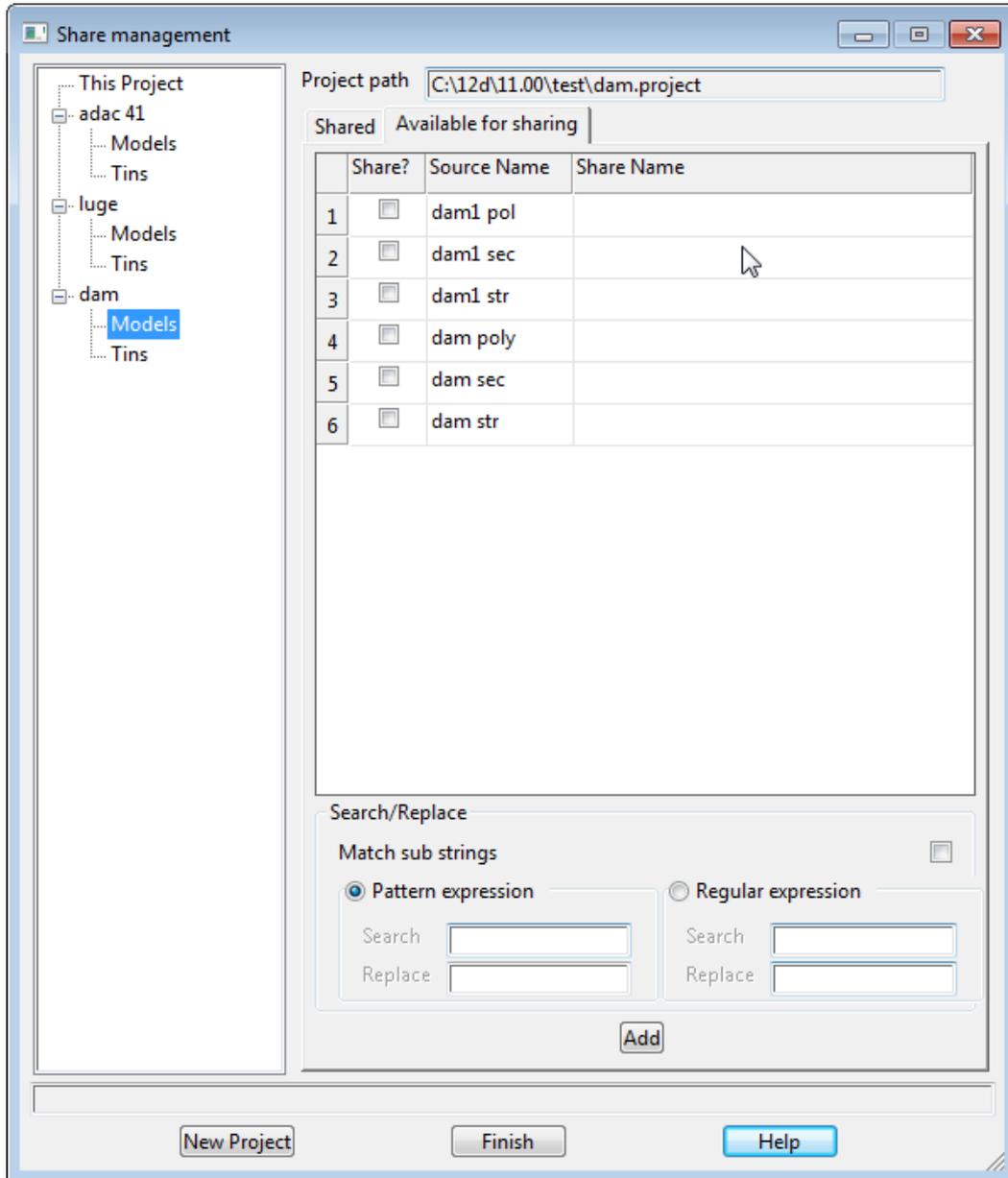
ticking in the **Share ?** column and then clicking the **Add** button.

### New Project Button

Clicking on the **New Project** button brings up the **Select a Project to Share From** panel.



Selecting a project and clicking on **Set** adds the given project as a new node in the **Share Management** panel and the **Model** and **Tin** nodes can be used to share in models and tins from that new project.



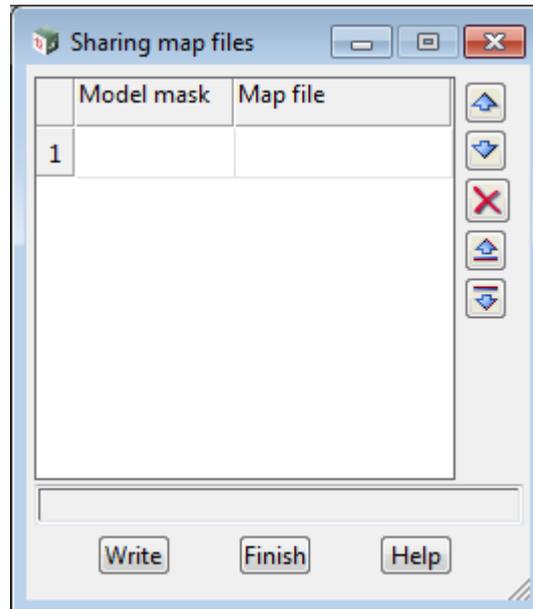
## 8.3 Sharing Map Files

The **Sharing Map Files** option allows different models **shared in** to use different **Map files**.

Clicking on the option

Project =>Management =>Sharing =>Sharing map files

brings up the **Sharing Map Files** panel.



### Model Mask-Map File grid:

The **Model Mask Map** grid is loaded from the file **sharing\_map\_files.4d** which is a **Set Up** file that is loaded when the project is opened.

The **Model Mask-Map File** grid consists of rows of text for **Model Mask** and one the same row, and associated **Map File**.

The **Model Mask** is a text string of alphanumeric characters and spaces, and including \* for a wild card and ? as a wild character.

For each **shared in** model, the model name is checked against the **Model Mask** in the first the first row, and if no match is made, is then tested against the next row. This is repeated until a match occurs.

When a match occurs, the **Map File** for the match row is applied to the model and no more tests against **Model Masks** are made.

If there is no match but there is a project **Sharing map file** (see the field **Projects >Sharing >Sharing map file** in the **Edit Environment Variables** panel), then the **Sharing map file** is applied to the **shared in** model.

If no match occurs and there is no project **Sharing map file** then the **shared in** model is left alone.

If the data in the **Model Mask Map** grid is modified then the modified file is written out with the **Write** button. A project restart is required for the new file to take effect.

### Write Button

When the **Write** button is selected, a **Write Setup File** panel comes up to specify where the **sharing\_map\_files.4d** file is to be written out to.

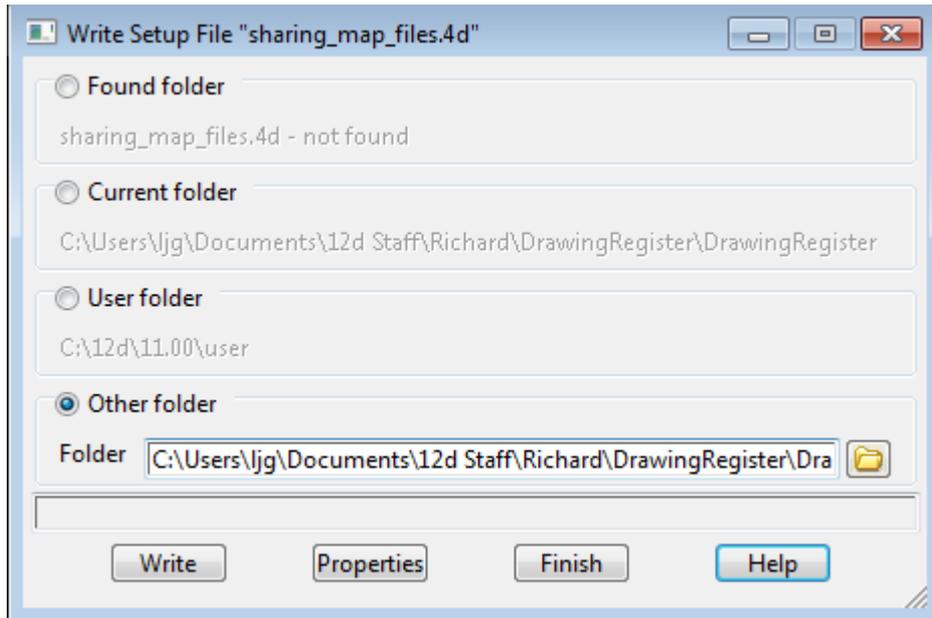
The choices on the panel allow the file to be written out to:

**Found folder** - the folder where the file currently being used by **12d Model** resides. This will be unavailable (greyed out) if the user doesn't have access to the folder.

**Current folder** - the folder where the project currently being used by **12d Model** resides. This will be unavailable (greyed out) if the user doesn't have access to the folder.

**User folder** - the **User** folder. This will be unavailable (greyed out) if the user doesn't have access to the folder.

**Other folder** - any folder can be selected.



A project restart is required for the new file to take effect

Note

When applying a **Map File** to a **shared in** model, only the **Basic** tab of the **Map File** is used and it is only used to change the colour of strings in the **shared in** model.

## 8.4 Localize Shares

**Position of option on menu:** Project =>Management =>Sharing =>Localize shares

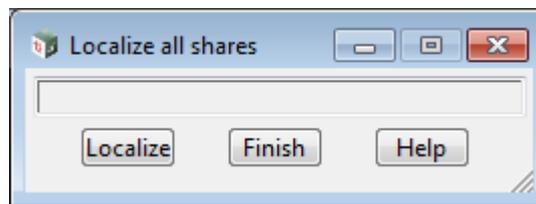
This option copies all the shared models and tins to the local project in a special folder and make them "local shares" so like the original shared models and tins, they still can't be edited.

This is useful for creating a project that is being sent to another user so that all the shared models and tins are included in the project.

### IMPORTANT WARNING

You can't go back to the original shares so make sure this is only run on a copy of a project and not a working project. If you only wanting copies of your shared models and tins in the project then use the Copy models and tins commands.

Selecting **Subscribe to share master files** brings up the **Subscribe to Sharing Master Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Localize</b>	button		
-----------------	--------	--	--

*creates "local" shares of all the shared models and tins.*

# 9. Smart Chainages

**Smart Chainages** began in V9 as **Start mode** and **End mode** replacing the **Start chainage** and **End chainage** boxes in the MTF Modifier commands which only supported a typed chainage value.

Many of the **Start** and **End modes** freed the commands from needing any typed chainage value and so things dynamically change with a change in data and geometry.

This continues the drive to **Chainage Free Design** which began with Computators in Super Alignments.

In V11, many new **modes** have been added for **Smart Chainages**, plus the **Smart Chainages** are starting to be used in options outside of the *MTF Modifier* commands.

Which Smart Chainage modes are available will depend on where the Smart Chainage is being used. For example, **Relative to Alias start** and **Relative to Alias end** and only applicable for Smart Chainages in grids where **Aliases** are allowed.

See

[9.1 Extension Value](#)

[9.2 Alias in Grids](#)

[9.3 Named Parts Accessing Vertical Parts](#)

[9.4 New Smart Chainage Modes](#)

## 9.1 Extension Value

For chainages given by **Start mode**, **End mode** or **Mode**, there is now a **Extension ref** field in the **Chainage** panel and the **Extension ref** value is **added** to the reference chainage calculated by the **Smart Chainage**.

In cases when the **Smart Chainage** first uses a chainage from another string, there can be an **Extension other** for the other string, and also an **Extension ref** for the reference string.

For example, for the **Smart Chainage Start of other string**, the value of **Extension other** is added to the *Start chainage* of a selected string and the point with that chainage is dropped onto the Reference string. The value of **Extension ref** is added to the chainage of the dropped point on the Reference string, to give the **final** chainage used.

The screenshot shows a dialog box with two sections: 'Start chainage' and 'End chainage'. Each section has a 'Mode' dropdown, a 'String' text field, and two 'Extension' fields (one for 'other' and one for 'ref'). The 'Start chainage' section has 'Start (other)' in the Mode dropdown, an empty String field, and empty Extension fields. The 'End chainage' section has 'Typed' in the Mode dropdown, '30' in the Chainage field, and an empty Extension ref field. There are icons for help and OK/Cancel on the right side of each section.

## 9.2 Alias in Grids

For MTF commands in a grid, the row can now be given a name called an **Alias**.

The **Alias** can be used in the **Smart Chainages** commands **Relative to Alias start** and **Relative to Alias end** when used in those grids.

For example, See [24.8 Alias for Left and Right Modifiers](#).

## 9.3 Named Parts Accessing Vertical Parts

The **Named Parts** fields for the Smart Chainage modes can now access the **Vertical named parts** as well as the **Horizontal named Parts** of a Super Alignment. The list contains **both** the horizontal and vertical named parts.

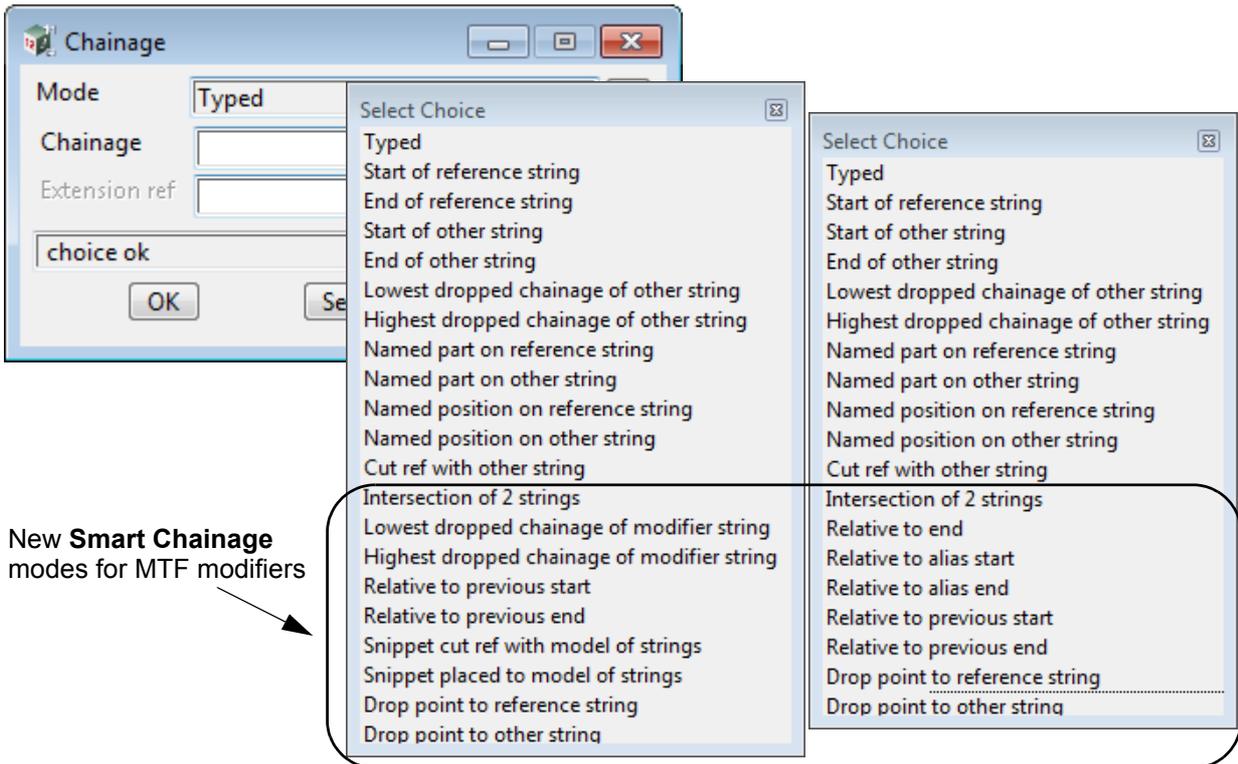
### Important Note

The Modifiers cannot distinguish between the names for horizontal and vertical named parts so **the horizontal names are given priority**. So if a vertical part has the same names as a horizontal part then the vertical name will not be shown in the named parts list.

## 9.4 New Smart Chainage Modes

There are new **Smart Chainage** modes in V11.

Which ones are available on the **Select Choice** pop up depends on which ones make sense for the particular panel field or grid field that the Smart Chainage is used in.



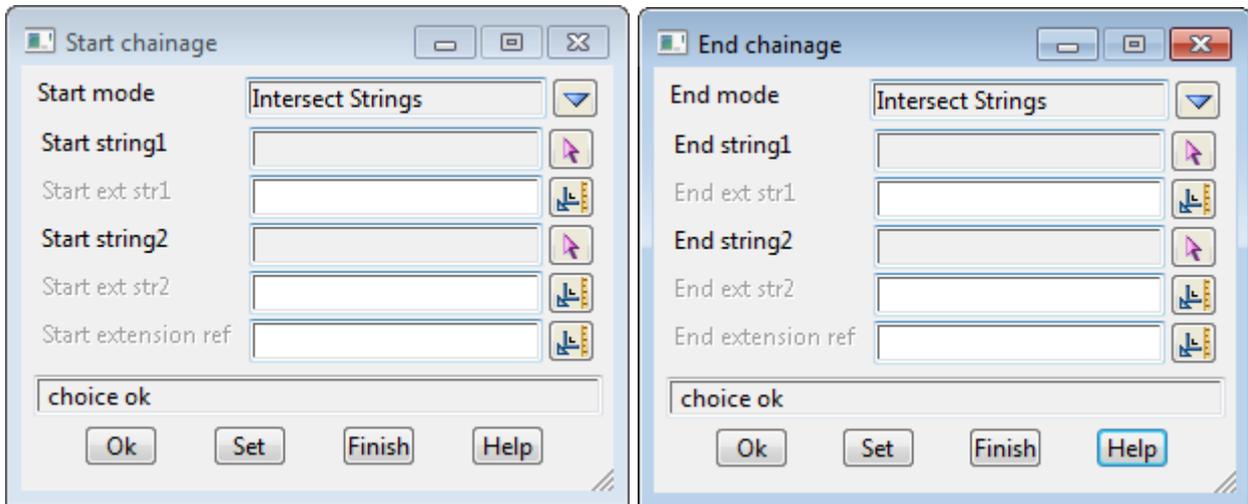
New Smart Chainage modes for MTF modifiers

See

- [9.4.1 Intersection of 2 strings](#)
- [9.4.3 Relative to Self End](#)
- [9.4.4 Relative to Self Start](#)
- [9.4.5 Relative to Alias Start](#)
- [9.4.6 Relative to Alias End](#)
- [9.4.7 Relative to Previous Start](#)
- [9.4.8 Relative to Previous End](#)
- [9.4.9 Drop Point to Reference String](#)
- [9.4.10 Drop Point to Other String](#)
- [9.4.11 Lowest Dropped Chainage of Modifier String](#)
- [9.4.12 Highest Dropped Chainage of Modifier String](#)
- [9.4.13 Snippet Cut Ref with Model of Strings](#)
- [9.4.14 Snippet Placed to Model of Strings](#)

## 9.4.1 Intersection of 2 strings

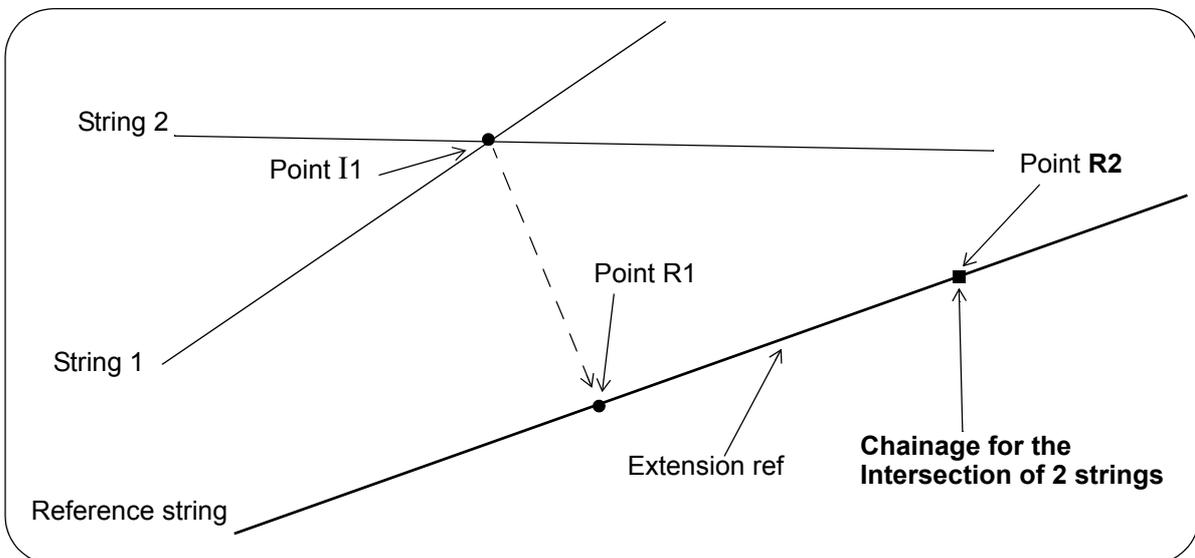
Selecting **Intersection of 2 strings** adds fields to select the two strings to intersect.



The *intersection* point **I1** of the two selected strings is calculated and dropped onto the Reference string.

The chainage value **Extension ref** is added to the chainage of the dropped point to give point **R2**.

The chainage for the Mode **Intersection of 2 strings** is the chainage of the point **R2**.



## 9.4.2 Intersection of 2 strings - Replaced by Simple Version

This option no longer exists. It has been replaced by a simpler version

Selecting **Intersection of 2 strings** adds fields to select the two strings to intersect.

First the *intersection* point **I1** of the two selected strings is calculated.

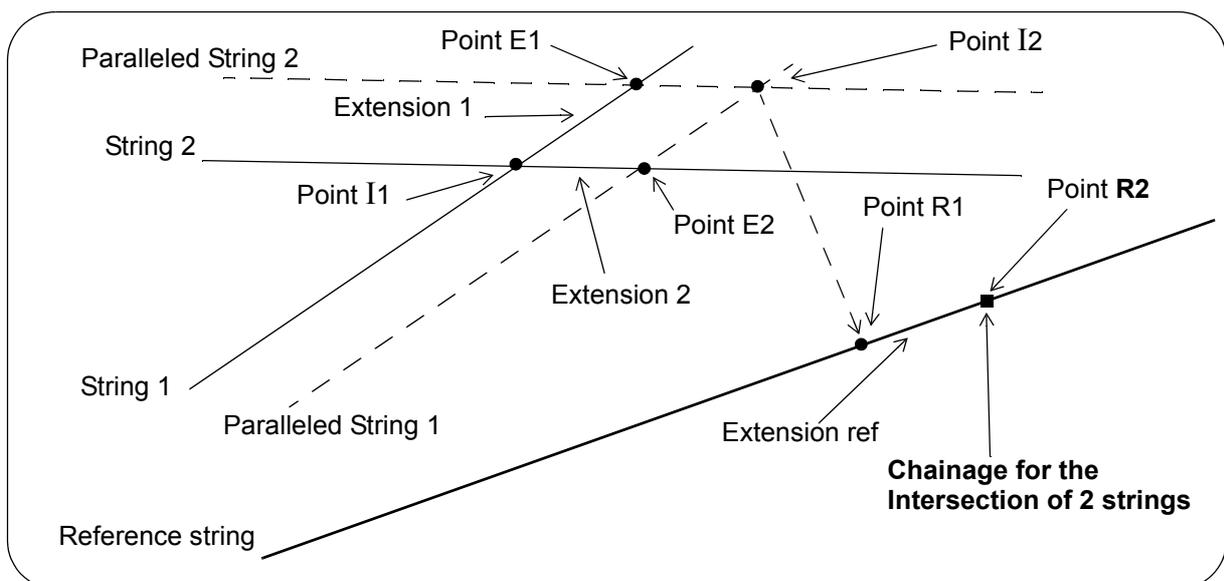
The chainage value **Extension 1** is added to the intersection point for **String 1** to give the point **E1**. **String 2** is paralleled so that the paralleled string passes through the point **E1**.

The chainage value **Extension 2** is added to the intersection point for **String 2** to give the point **E2**. **String 1** is paralleled so that the paralleled string passes through the point **E2**.

The two paralleled strings are then intersected and the intersection point **I2** then dropped onto the Reference string (point **R1**).

The chainage value **Extension ref** is added to the chainage of the dropped point to give point **R2**.

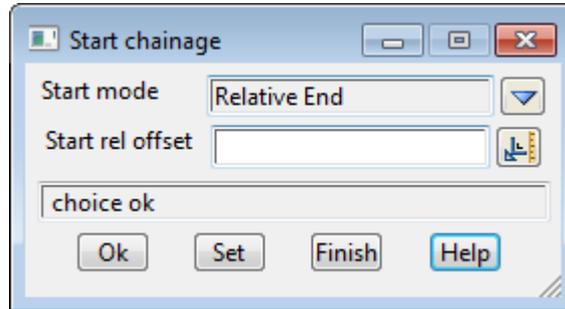
The chainage for the Mode **Intersection of 2 strings** is the chainage of the point **R2**.



### 9.4.3 Relative to Self End

For MTF Modifiers Grid.

For **Start chainage mode**, selecting **Relative to self end** gives



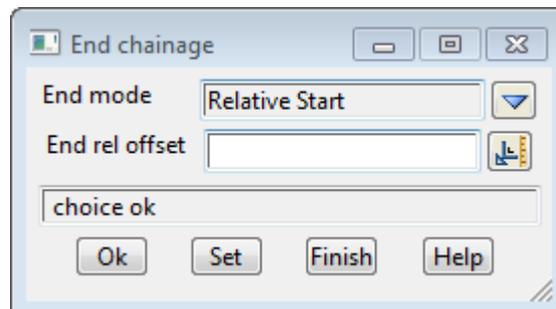
**Extension ref** is **added** to the end chainage for the command.

This is useful when you know the end chainage of the current command and want the start chainage of the current command to be **relative to the end chainage** of the **current command**.

### 9.4.4 Relative to Self Start

For MTF Modifiers Grid.

For **End chainage mode**, selecting **Relative to self start** gives



**Extension ref** is **added** to the start chainage of this command.

This is useful when you know the start chainage of the current command and want the end chainage of the current commands to be **relative to the start chainage** of the **current command**.

### 9.4.5 Relative to Alias Start

For MTF Modifiers Grid.

**Start alias** uses the Alias of the row of the *MTF Modifiers* to take the start chainage from. See [24.8 Alias for Left and Right Modifiers](#).

**Extension ref** is **added** to the start chainage of the row with the alias given in **Start alias**.

This is useful when you know the start chainage of the row with the given alias and want the chainage of the current command to be **relative to the start** of the **alias command**.

### 9.4.6 Relative to Alias End

For MTF Modifiers Grid.

**End alias** uses the Alias of the row of the *MTF Modifiers* to take the end chainage from. See [24.8 Alias for Left and Right Modifiers](#).

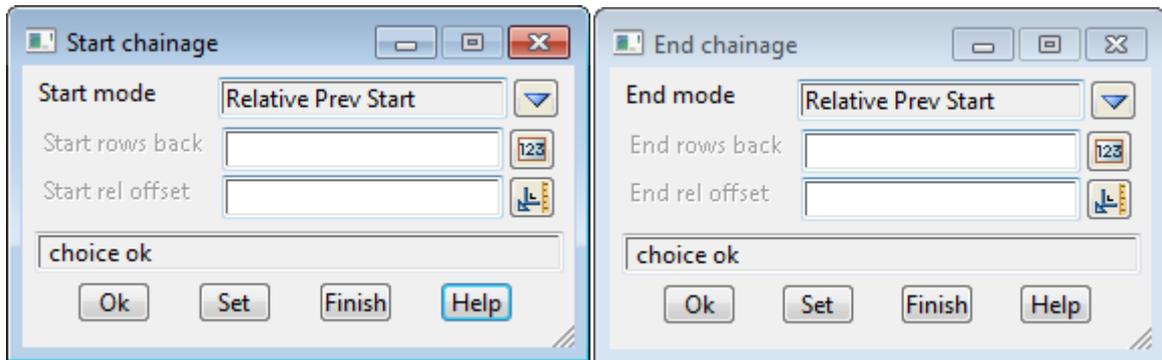
**Extension ref** is **added** to the end chainage of the row with the alias given in **End alias**.

This is useful when you know the end chainage of the row with the given alias and want the chainage of the current command to be **relative to the end** of the **alias command**.

### 9.4.7 Relative to Previous Start

For MTF Modifiers Grid.

For **Start/End chainage**, selecting **Relative to previous start** gives



**Rows back** gives the number of rows in the *MTF Modifiers* grid to go back to get the **Start** chainage from. The value must be **one** or greater.

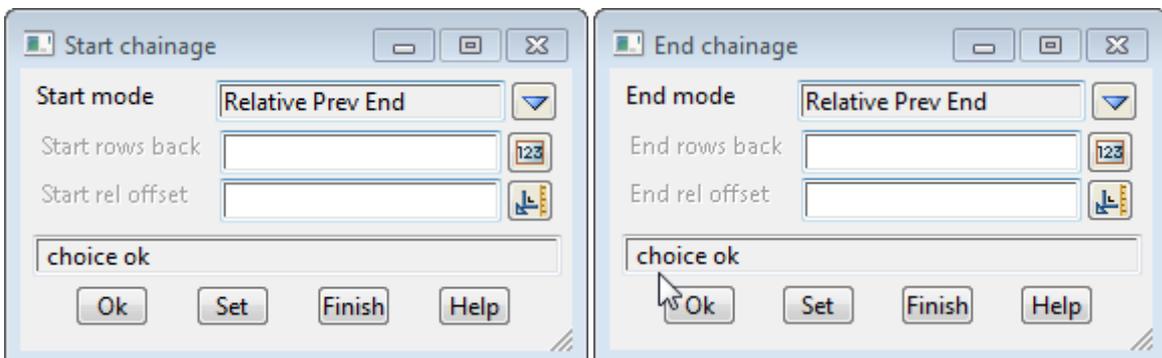
**Extension ref** is added to the **Start** chainage of the selected row.

This is useful when you know the start chainage of an earlier command and want the chainage of the current command to be relative to the start chainage of the earlier command.

### 9.4.8 Relative to Previous End

For MTF Modifiers Grid.

For **Start/End chainage**, selecting **Relative to previous end** gives



**Rows back** gives the number of rows in the MTF grid to go back to get the **End** chainage from. The value must be **one** or greater

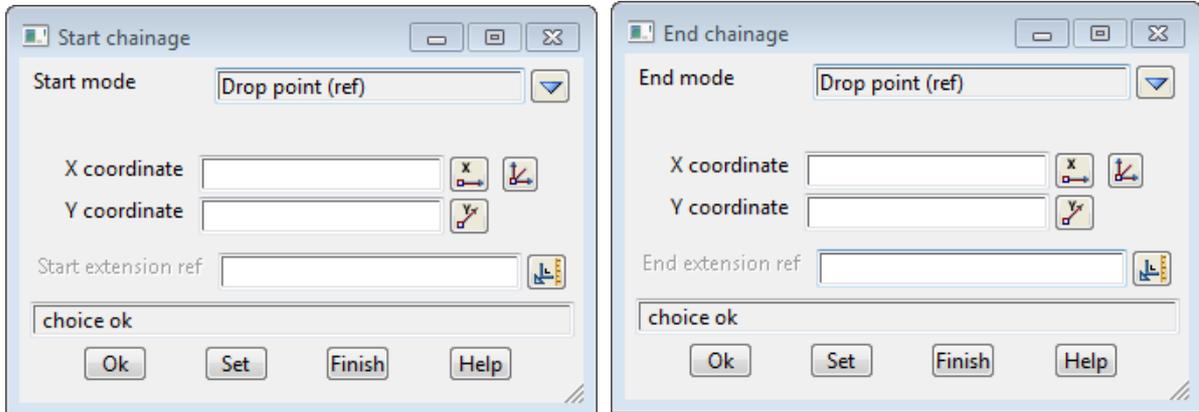
**Extension ref** is **added** to the **End** chainage of the selected row.

This is useful when you know the end chainage of an earlier command and want the chainage of

the current command to be relative to the end chainage of the earlier command.

### 9.4.9 Drop Point to Reference String

For **Start/End chainage mode**, selecting **Drop point to reference string** gives

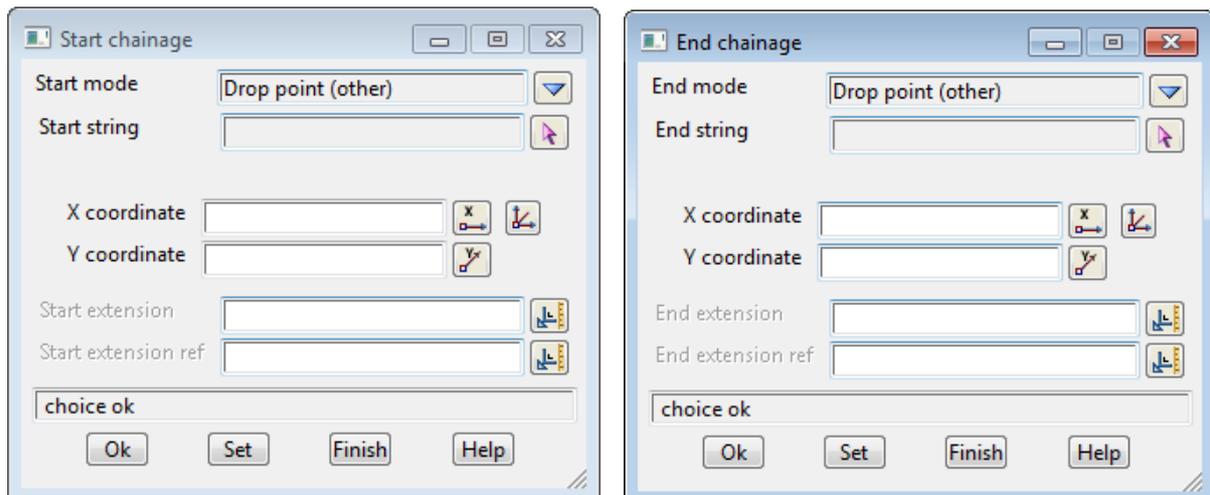


**X/Y coordinate** is the coordinate of the point to drop onto the reference string to give a chainage. **Extension ref** is added to the dropped chainage to give the Start/End chainage for the current command.

This is useful when you know the coordinates of a point that you want the Start/End chainage of that point dropped onto the Reference string.

### 9.4.10 Drop Point to Other String

For **Start/End chainage mode**, selecting **Drop point to other string** gives



**Other string** is a selected string.

**X/Y coordinate** is the coordinate of the point that is dropped onto the selected other string.

**Extension other** is added to the dropped chainage on the selected string. The point at this chainage is then dropped onto the reference string.

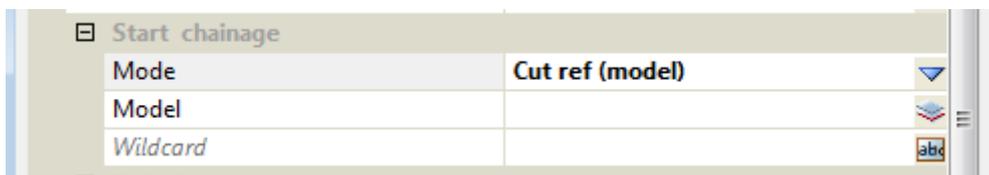
**Extension ref** is added to the point dropped on the reference string to give the **final** chainage to This is useful when you know the coordinates of a point that you want to drop onto another string, which is then dropped onto the reference string.

### 9.4.11 Lowest Dropped Chainage of Modifier String

### 9.4.12 Highest Dropped Chainage of Modifier String

### 9.4.13 Snippet Cut Ref with Model of Strings

You can have a model of strings and the snippet is applied where the **Start chainage** of the snippet is the Reference string cut with each of the strings in the model. The strings to use can be restricted by a using wild card.

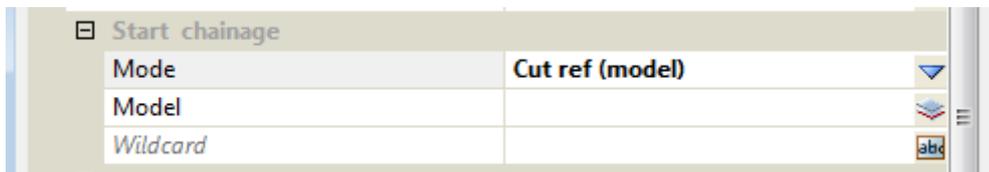


**Model**                                      model box                                      available models  
*the model of strings that the snippet will be applied to where the reference string cuts the string.*

**Wildcard**                                      text box  
*if **not blank**, only those strings in the model **Model** that satisfy **Wildcard** are used.  
 If **blank**, all the strings in the model **Model** are used.*

### 9.4.14 Snippet Placed to Model of Strings

You can have a model of strings and the snippet is applied to each string in the model with the **Start chainage** of the snippet taken as the beginning of the string dropped onto the Reference string and the **End chainage** of the snippet taken as the end of the string dropped onto the reference string. The strings to use can be restricted by a using wild card.



**Model**                                      model box                                      available models  
*the model of strings that the snippet will be applied to using the start and end of each string as the Start and End chainages for the snippet.*

**Wildcard**                                      text box  
*if **not blank**, only those strings in the model **Model** that satisfy the wildcard are used.  
 If **blank**, all the strings in of strings in the model **Model** are used.*

# 10. Chains

See

- [10.1 Some Options That Didn't Record & Now Can](#)
- [10.2 Warning Prompt Tick Box](#)
- [10.3 Command Name Moved](#)
- [10.4 Icons for Activate and Deactivate](#)
- [10.5 Clicking on a Resolve Command Highlights the SA](#)
- [10.6 You Get a Warning When Leaving a Project and a Chain is Still Open](#)
- [10.7 Clicking Command Name Icon Regenerates Name](#)
- [10.8 Re-Recording Does Not Change the Command Name](#)
- [10.9 There is a Preview Button for Functions](#)
- [10.10 Recording an Apply MTF](#)
- [10.11 Disable Delete Confirmation](#)
- [10.12 Copy and Paste Between Chains](#)
- [10.13 Chain View Commands](#)
- [10.14 Creating a View Giving Top Left and Bottom Right](#)
- [10.15 Dump Image of a View](#)
- [10.16 Delete Function Command](#)
- [10.17 Delete All Functions Command](#)
- [10.18 Prompt Command](#)
- [10.19 Output Window Commands](#)
- [10.20 Sleep Command](#)
- [10.21 Save Project](#)
- [10.22 Copy Command](#)
- [10.23 Dump Image of SLX of a Panel](#)
- [10.25 Command - If Function Exists](#)
- [10.26 Command - If Parameter Equals - Parameter Pop Up](#)
- [10.27 Commands for Multiple Lines](#)
- [10.28 Chain Parameters Editor](#)
- [10.29 Chain Parameters as a 12dPL Arguments](#)
- [10.30 Printing the Value of a Chain Parameter](#)
- [10.31 Append for PPF Grid Parameters](#)
- [10.32 PVF Substitution in User Menus and Toolbars](#)

## 10.1 Some Options That Didn't Record & Now Can

File =>Templates =>Templates output =>One template

File =>Templates =>Templates output =>All templates

Design =>Apply =>Apply MTF

String =>Label =>Cut/Fill =>Tadpoles

String =>Label =>Cut/Fill =>Ticks

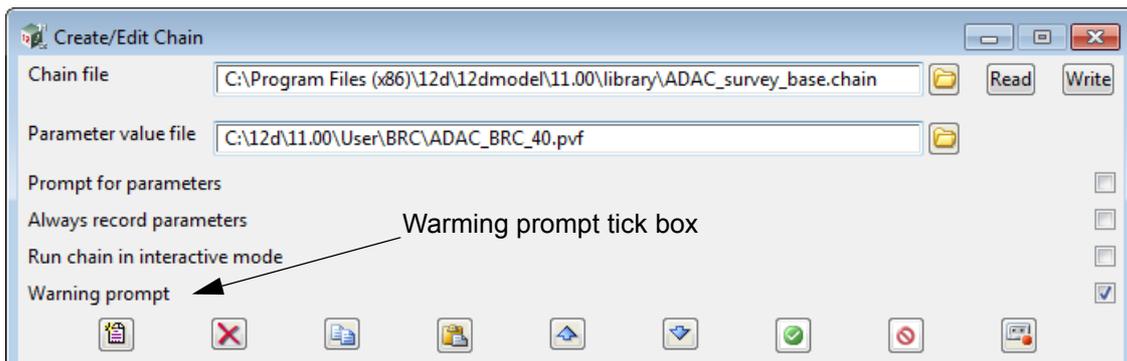
## 10.2 Warning Prompt Tick Box

When the **Warning prompt** is not ticked, validations do not occur straight away.

For example, when the **Warning prompt** is not ticked, you can hit **Insert** many times and you won't get the panel

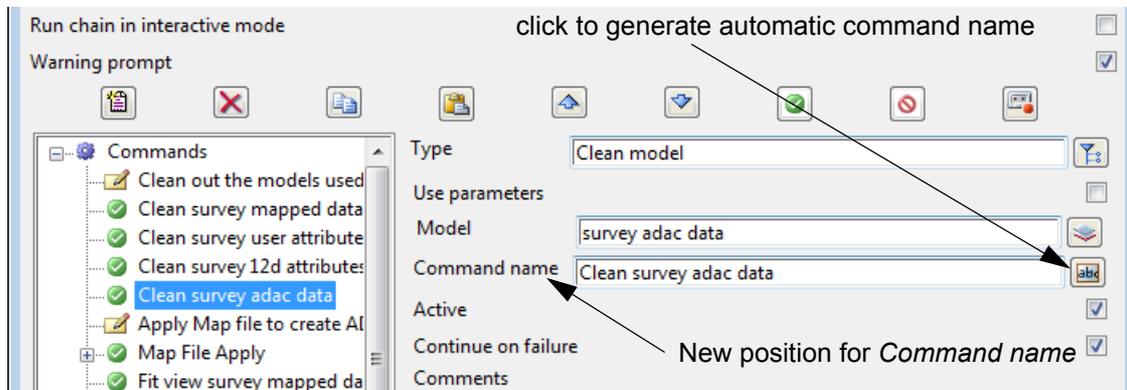
"Some changes you have made are invalid. Do you wish to change nodes and discard your changes?".

When the **Warning prompt** is not ticked, a confirmation prompt does not come up when you delete commands.



## 10.3 Command Name Moved

The field **Command name** is now further down the panel so the information required to create an automatic name occurs and the automatic name is placed in the **Command name** field *before* you get down to type into the **Command name** field. Most of the time the automatic name is sufficient and nothing else is needed in the **Command name** field.



When you rerecord a command, the **Command name** is no longer modified. But clicking on the text icon at the end of the **Command name** field will generate the automatic **Command name** and write it to the **Command name** field.

## 10.4 Icons for Activate and Deactivate

There are new icons on the Chain panel to **Activate** and **Deactivate** the selected Chain Command.



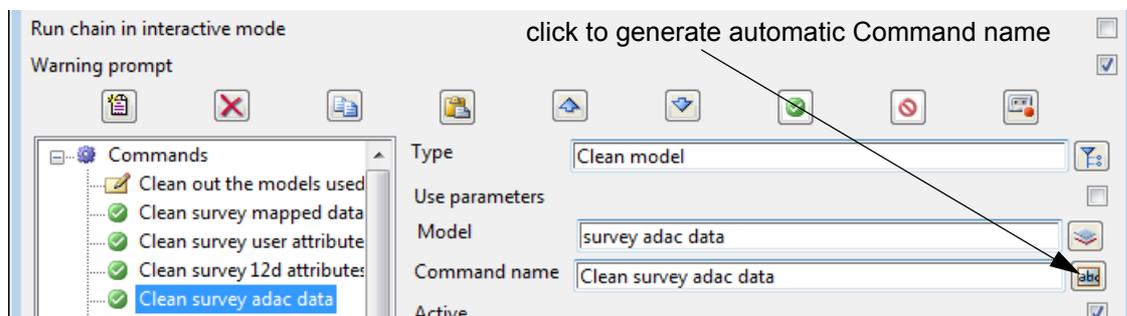
## 10.5 Clicking on a Resolve Command Highlights the SA

This went missing in **12d Model 10** but has been put back in.

## 10.6 You Get a Warning When Leaving a Project and a Chain is Still Open

## 10.7 Clicking Command Name Icon Regenerates Name

Clicking on the **abc** (text) icon at the end of the **Command name** field will generate the automatic **Command name** and write it to the **Command name** field.



## 10.8 Re-Recording Does Not Change the Command Name

When you rerecord a command, the **Command name** is no longer modified.

But clicking on the text icon at the end of the **Command name** field will regenerate the automatic **Command name** and write it to the **Command name** field (see [10.7 Clicking Command Name Icon Regenerates Name](#)).

## 10.9 There is a Preview Button for Functions

## 10.10 Recording an Apply MTF

An **Apply MTF** can now be recorded in a chain.

## 10.11 Disable Delete Confirmation

Message box now offers a check box that says: "Don't ask again this session." Ticking this off will turn off confirmations for the current run of **12d Model**.

You can also set an environment variable, ASK\_ON\_CHAIN\_COMMAND\_DELETE\_4D 0 to

permanently stop it asking.

This ENV variable is also available under **env.4d >GUI >Chain Editor >Prompt** for deleting on a chain command

This way, the default (or undefined behaviour) is ON

## 10.12 Copy and Paste Between Chains

A command can be copied and pasted between two chains.

## 10.13 Chain View Commands

There are new View Chain commands:

**Views >Maximize view** - maximise the given view.

**Views >Minimize view** - minimise the given view.

**Views >Restore view** - restore the given view.

**Views >Redraw a view** - redraw a single view.

**Views >Change view colour** - changes the background colour for a given view.

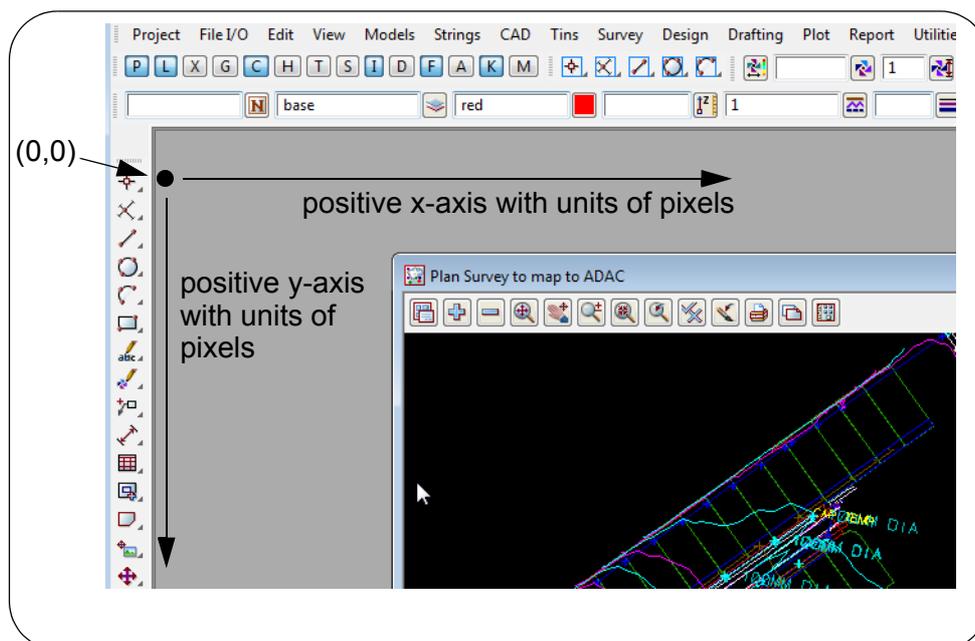
## 10.14 Creating a View Giving Top Left and Bottom Right

The Chain command

**Views > Create view**

now allows you to give the Top Left and Bottom Right to position and size the view.

The coordinate system has (0,0) at the Top Left of the **12d Model** View area, and the X-axis going across the screen and the Y-axis going down the screen. The units are pixels.

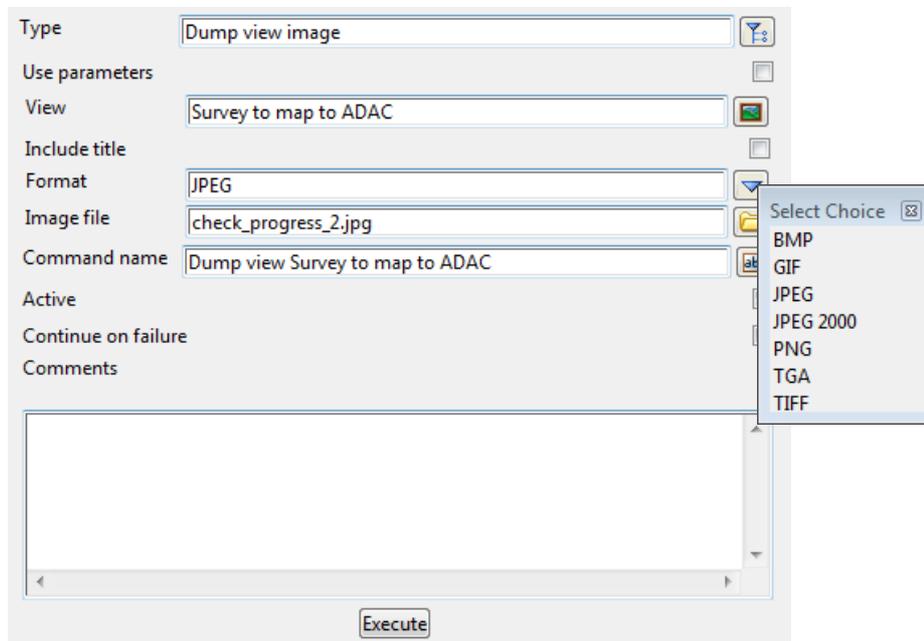


## 10.15 Dump Image of a View

The Chain command

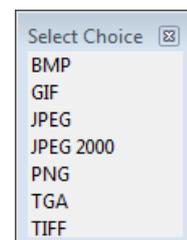
### **Views >Dump view image**

write out an image of the given view.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Type</b> <i>the chain command</i>	chain command box		available chain commands
<b>User parameters</b>	tick box		
<b>View</b> <i>name of the view to write out an image of file of.</i>	View box		available views
<b>Include title</b> <i>if <b>ticked</b>, the title of the view and the icons on the view are included in the image. If <b>not ticked</b>, the title of the view and the icons on the view are <b>not</b> included in the image.</i>	tick box		
<b>Format</b>	choice box		



*image format to use.*

**Image file** file box  
*the image is written out to this file.*

**Command name** text box

*the name to give the command - this is what is displayed in the Chain list.*

**Active** tick box

*if **ticked** then the command is run when the Chain is run.*

*If **not ticked** then the command is not run when the Chain is run.*

**Continue on failure** tick box

*if **ticked** and there is an error when running the command, processing continues to the next command in the Chain.*

*If **not ticked** and there is an error when running the command then the Chain terminates.*

**Comments** text box

*any information typed into this box is kept as a Comment for this Chain command.*

**Execute** button

*run the current chain Command.*

## 10.16 Delete Function Command

The Chain command

**Execution >Delete function**

deletes a function.

It will optionally *clean* all the strings created by the function.

## 10.17 Delete All Functions Command

The Chain command

**Execution >Delete all functions**

deletes all functions.

It will optionally *clean* all the strings created by the functions.

## 10.18 Prompt Command

### 10.18.1 Prompt Title

In the Chain command

**Other >Prompt**

there is now a **Prompt** title box and when not blank, it is used as the Title on the top of the **Prompt** panel.

### 10.18.2 Prompt Coming Up at the Same Place

With the Chain command

**Other >Prompt**

the *Prompt* panel now comes up at the same place each time. The position is not under user control.

## 10.19 Output Window Commands

The Chain command

**Other >Clear output window**

clears the Output Window

The Chain command

**Other >Write to output window**

write a one line message to the Output Window

## 10.20 Sleep Command

The Chain command

**Other >Sleep**

puts the chain to sleep for a user given number of seconds

## 10.21 Save Project

The Chain command

**Other >Save project**

saves the current project to disk.

## 10.22 Copy Command

The Chain command

**Files >Copy file**

copies a file to a user given path and name.

The screenshot shows a configuration dialog for the 'Copy file' command. It includes the following fields and options:

- Type:** Copy file
- Use parameters:**
- Original File:** final.rpt
- New File:** C:\12d Jobs\airport\final.rpt
- Command name:** Copy file final.rpt
- Active:**
- Continue on failure:**
- Comments:** (empty)

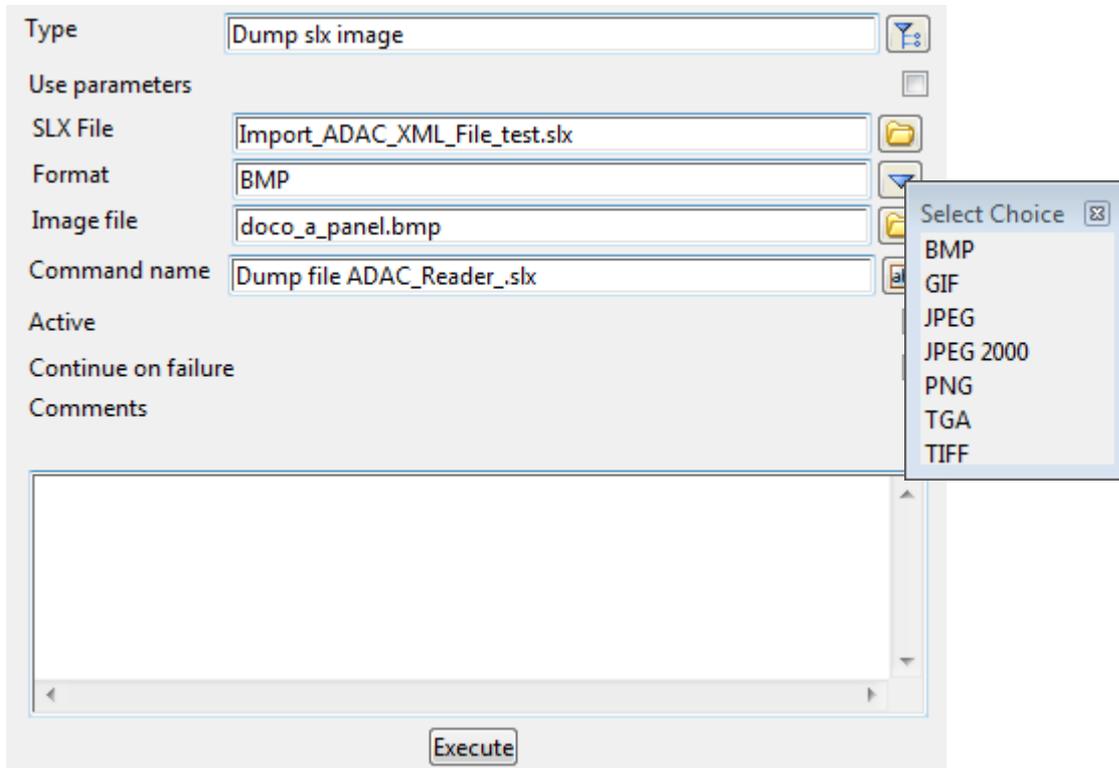
The file being copied will overwrite an existing file.

## 10.23 Dump Image of SLX of a Panel

The Chain command

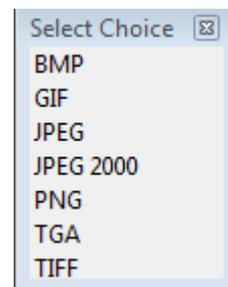
**Other >Dump panel image**

writes out an image of the panel that the SLX creates.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Type</b> <i>the chain command</i>	chain command box		available chain commands
<b>User parameters</b>	tick box		
<b>SLX File</b> <i>name of the slx file to read in and display the panel it describes, and then write out an image for:</i>	file box		available *.slx commands
<b>Format</b>	choice box		



*image format to use.*

**Image file** file box

*the image is written out to this file.*

**Command name**                      text box

*the name to give the command - this is what is displayed in the Chain list.*

**Active**                                      tick box

*if **ticked** then the command is run when the Chain is run.*

*If **not ticked** then the command is not run when the Chain is run.*

**Continue on failure**                      tick box

*if **ticked** and there is an error when running the command, processing continues to the next command in the Chain.*

*If **not ticked** and there is an error when running the command then the Chain terminates.*

**Comments**                                      text box

*any information typed into this box is kept as a Comment for this Chain command.*

**Execute**                                      button

*run the current chain Command*

## 10.24 Shell Command Now has Separate Argument Line

The Chain command

### **Execution >Shell Command**

now has an **Executable** and a separate **Arguments** fields.

Type	Shell command	
Executable		
Arguments		
Command name	New	
Active		<input checked="" type="checkbox"/>
Continue on failure		<input type="checkbox"/>
Comments		

Also the value of a Chain parameter can be used as an argument passed to a **Executable** program.

The parameter name is enclosed in square brackets ([ and ]) and then surrounded by double quotes ("")

"[parameter\_name]"

and this is entered in the **Arguments** field in the place of the argument of the **Executable** that it replaces.

## 10.25 Command - If Function Exists

The Chain command

### **Conditionals >If Function Exists**

tests if a given function exists.

## 10.26 Command - If Parameter Equals - Parameter Pop Up

For the Chain command

### **Conditionals >If Parameter Equals**

clicking on the **abc** icon at the end of the **Parameter name** field will bring up a list of all the parameters defined in the **Parameter value file**.

## 10.27 Commands for Multiple Lines

In *12d Model 11*, you can select more than one line of a Chain by using the standard Microsoft Ctrl> LB etc to select and highlight a number of lines of the Chain.

When multiple lines of the Chain are selected, the following icons work for all the selected lines.

Delete

Make Active/Inactive

Move up, Move Down,

Copy, Paste to after the selected command

Paste after the selected command in another Chain

Note that <Ctrl>+c and <Ctrl>+v cannot be used for Copy and Paste. Only the Copy and Paste icons can be used for **Copy** and **Paste**.

## 10.28 Chain Parameters Editor

See

[10.28.1 Icons for Parameter Types](#)

[10.28.2 Comments](#)

[10.28.3 Search](#)

[10.28.4 Set No Longer Needed](#)

### 10.28.1 Icons for Parameter Types

There are now icons on the left of the parameter node showing the type of parameter. For example, Real, Text, tin.

### 10.28.2 Comments

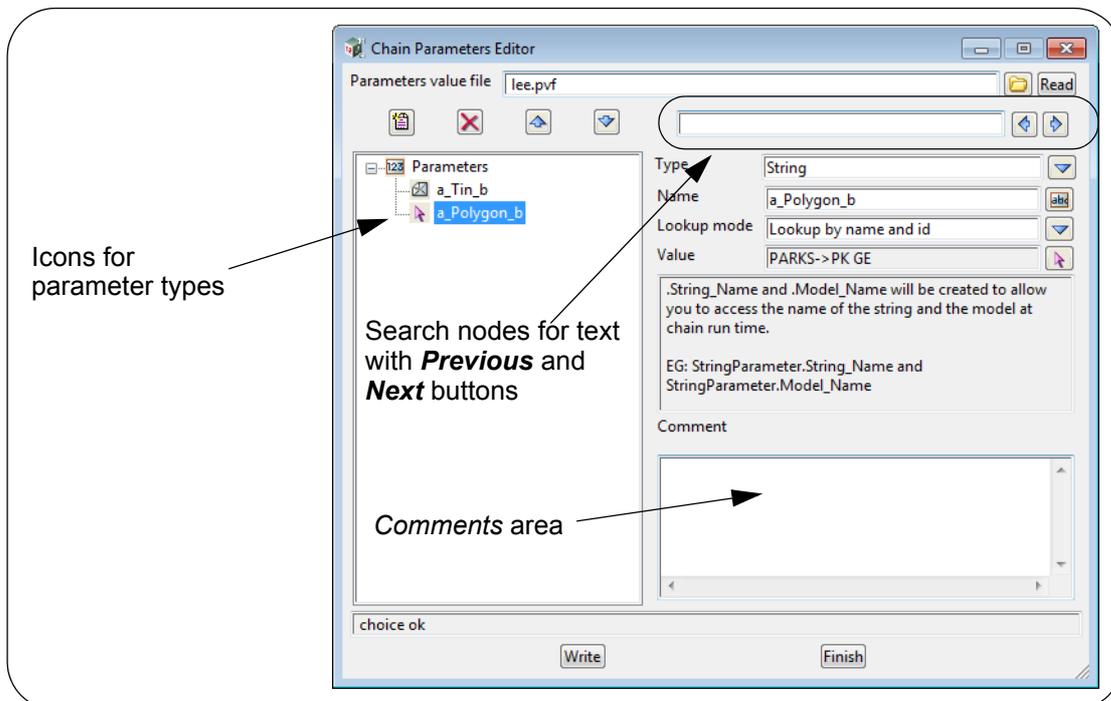
You can now enter *Comments* for a Parameter.

### 10.28.3 Search

You can now enter text to find a match in the node names, with **Next** and **Previous** to find multiple occurrences.

### 10.28.4 *Set* No Longer Needed

When you make changes you no longer have to click on **Set** to see the changes. The **Set** button has been removed.



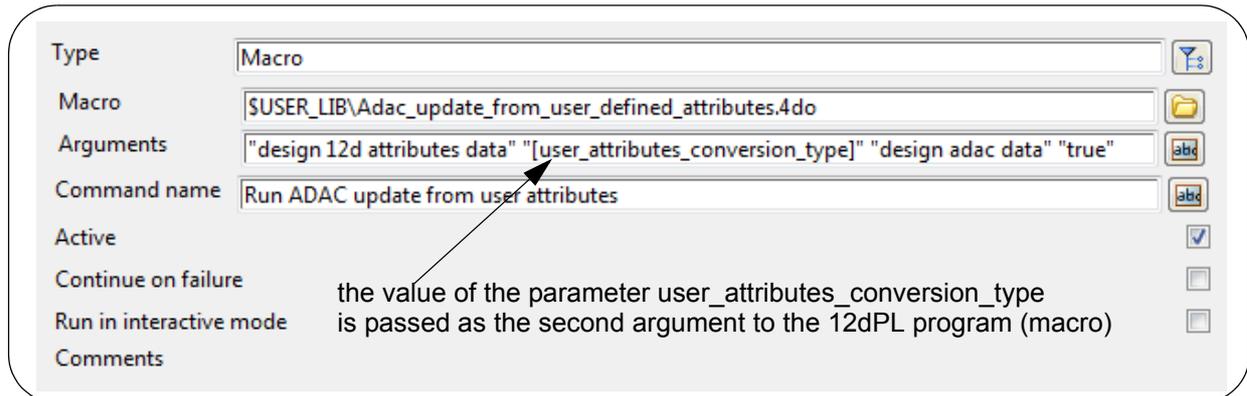
## 10.29 Chain Parameters as a 12dPL Arguments

The value of a Chain parameter can be used as an argument passed to a 12dPL program (also known as 12PL's or **12d Model** macros).

The parameter name is enclosed in square brackets ([ and ]) and then surrounded by double quotes ("")

"[parameter\_name]"

and this is entered in the **Arguments** field in the place of the argument that it replaces.



## 10.30 Printing the Value of a Chain Parameter

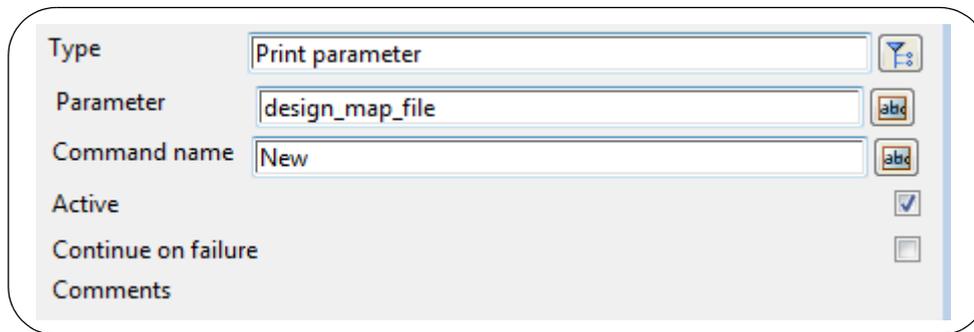
The Chain command

**Other >Print parameter**

prints to the *Output Window*

- (a) the name of the parameter
- (b) the type of the type
- (c) the value of the parameters where the *Print parameter* command is processed in the Chain in the format

*parameter\_name* [type:parameter\_type] = parameter\_value



As an example, for the parameter *design\_map\_file*, this would print to the **Output Window**:

---

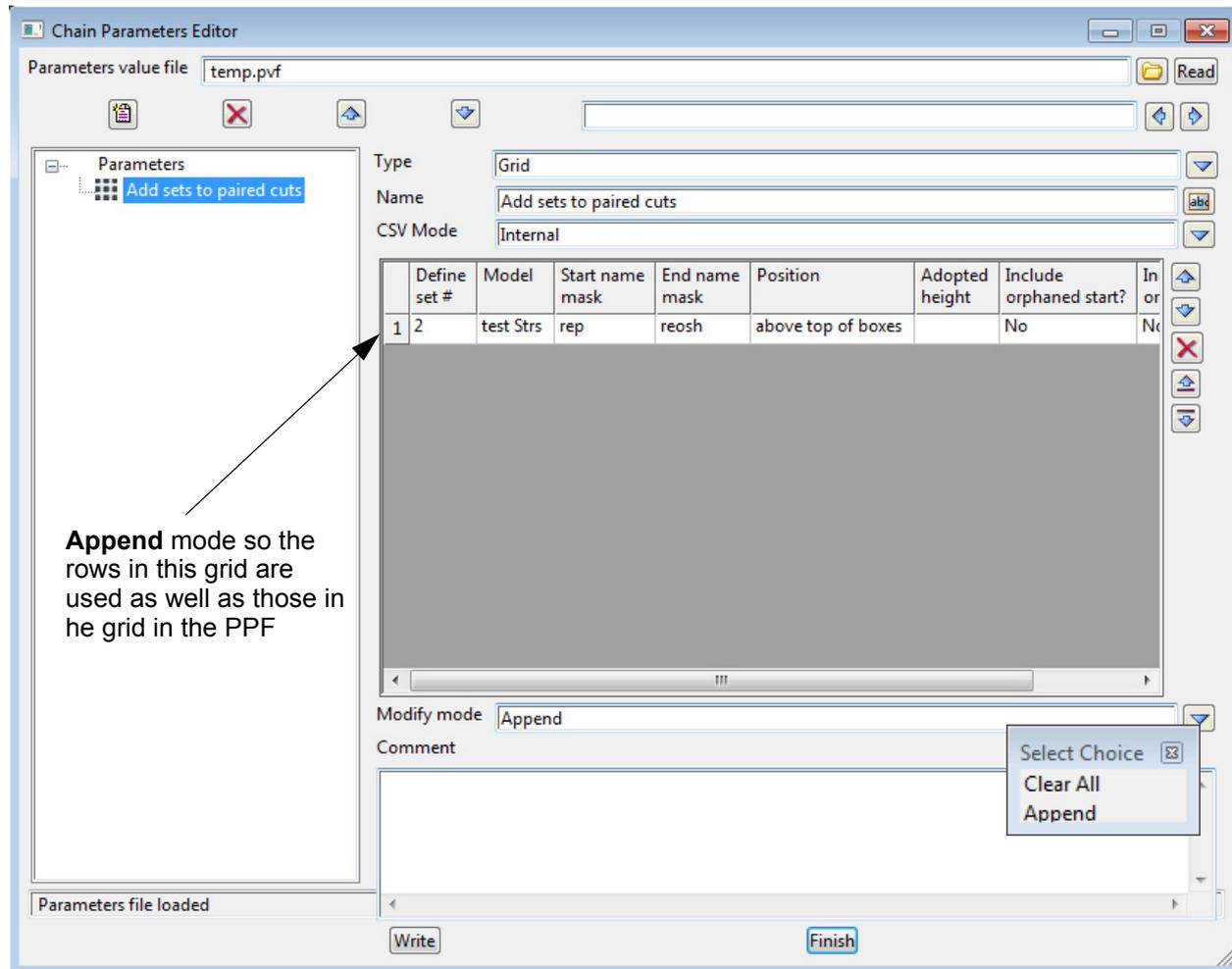
```
Starting chain: ADAC_design_base.chain
Executing print parameter command: New
design_map_file (type: Text) = $USER_LIB\ADAC_GCCC_Map_Design_to_ADAC_40.mapfile
Executing Comment command: Clean out the models used in processing
Executing Clean model command: Clean design mapped data
```

## 10.31 Append for PPF Grid Parameters

When you have parameters for a grid in a PPF file, there is a **Modify mode** with choices **Clear All** or **Append**.

**Clear All** Does what V10 does. That is, ignores all the current rows in the grid in the PPF and only uses the rows in the Parameter grid.

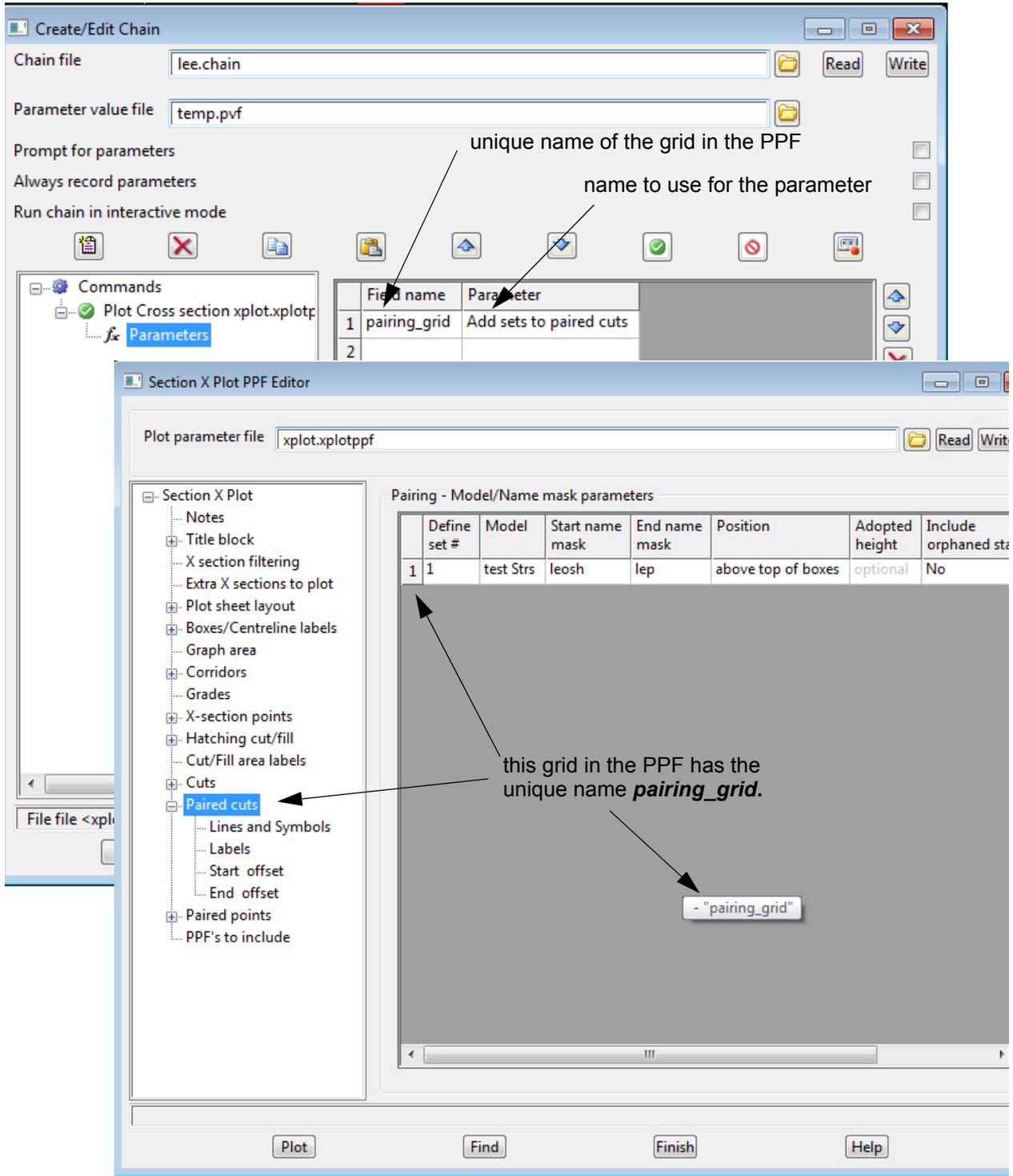
**Append** Uses all the current rows in the grid in the PPF plus the rows in the Parameter grid.



This is used in a Chain for plotting a PPF which has parameters for the grid called **pairing\_grid**.

**Note** - each grid in a PPF has a unique name and that name can be found by simply moving the cursor over the grid area of the PPF.

As an example of using the parameter file in a chain plotting the *Section X Plot PPF* **xplot.xplotppf**.



## 10.32 PVF Substitution in User Menus and Toolbars

In V10 a Chain can be run from a user menu or a user toolbar by using the **Command**:

```
Command "chain chain_file_name"
```

where *chain\_file\_name* is the name of the chain.

In **V11** it is possible to give the name of the pvf (parameter value file) to use for the chain *chain\_file\_name* in place of the pvf mentioned in the chain *chain\_file\_name* itself.

The command is

```
Command "chain -pvf pvf_file_name chain_file_name"
```

where *pvf\_file\_name* is the name of the pvf file to now use with the chain *chain\_file\_name*.

Note that the

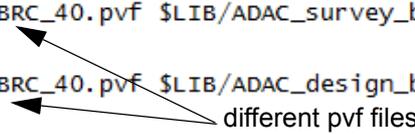
```
-pvf pvf_file_name
```

is still inside the double quotes (").

Using the *-pvf* means that the one base chain can be easily used in menus and toolbars but with different pvf files.

For example, the chain *ADAC\_Survey\_base.chain* occurs many times in the "ADAC Survey 4.0 Chains" menu but each time it has a different pvf file.

```
Menu "ADAC_BRC 4.0 Chains" {
  Button "BRC Survey to ADAC 40 chain" {
    Command "chain -pvf $USER/BRC/ADAC_BRC_40.pvf $LIB/ADAC_survey_base.chain"
  }
  Button "BRC Design to ADAC 40 chain" {
    Command "chain -pvf $USER/BRC/ADAC_BRC_40.pvf $LIB/ADAC_design_base.chain"
  }
}
```



**Same chain *ADAC\_survey\_base.chain* Run but With Different pvf Files**

# 11 BIM

See

[11.1 What is BIM ?](#)

[11.2 The 12d Approach to BIM](#)

[11.3 The UK Government and BIM](#)

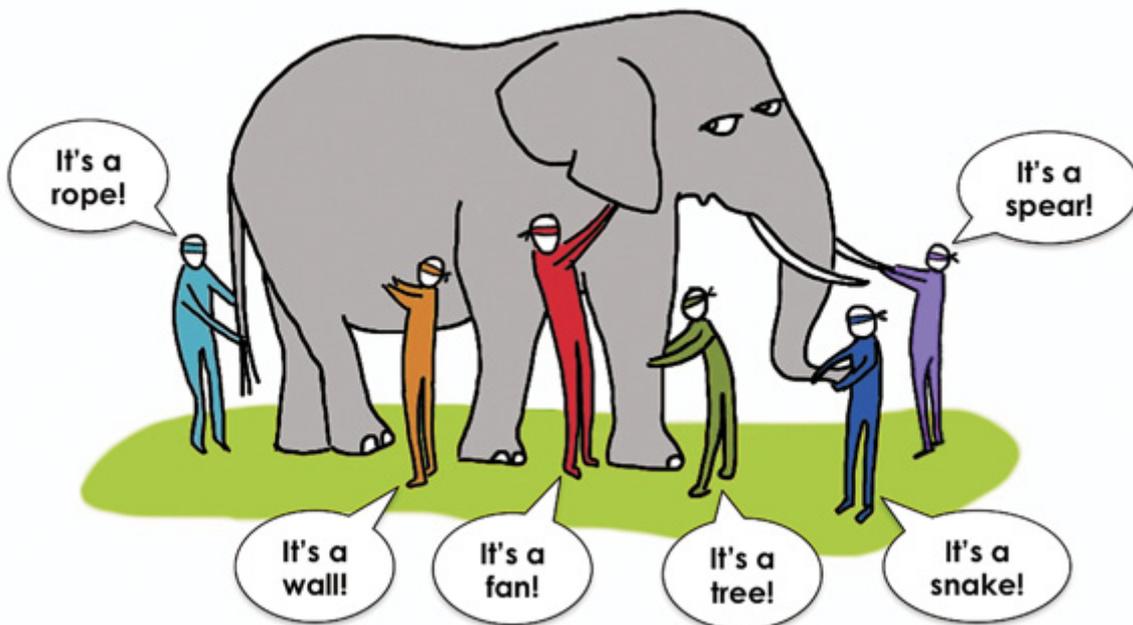
[11.4 Common Terms in BIM](#)

[11.5 What is Wrong with Local Coordinates ?](#)

[11.6 IFCs](#)

## 11.1 What is BIM ?

What BIM is seems to depend on who you are talking to. And the various participants in the design, construction and management of the asset (building or infrastructure) also emphasise different aspects of BIM.



If a wide search of the literature for BIM and ISO Standards is made, and all the marketing fluff is ignored, one quickly sees that almost universally the **BI** in BIM is **Building Information** and although BIM is about the process, all the objects are for a Building on a site. The other constant is that the International Standard for transferring BIM data is the vendor independent format of IFCs. For more information on IFCs, see [15.2.7 IFC Output](#).

In 2011 the BIM movement received a major boost when the UK Government released the **Government Construction Strategy** which aimed to reduce the cost of government construction projects by 15-20% and BIM was identified as one of the strategies to achieve help the savings.

The **Strategy** contained an **Action Plan** that included the objective for BIM to *introduce a*

*progressive programme of mandated use of fully collaborative Building Information Modelling for Government projects by 2016.*

One early outcome of the **Strategy** was for BuildingSMART to develop a national BIM standard with the principles of interoperability. This will be key to future delivery of Level 3 "Open & Shared" BIM.

**buildingSMART** is now the International group looking after the development of the **Industry Foundation Classes** (IFCs) which is the data model for the Open Standard for BIM. See <http://www.buildingsmart-tech.org>.

For more information, see [11.3 The UK Government and BIM](#) and for some common BIM terms and the **Levels of BIM** outlined by the UK Government, see [11.4 Common Terms in BIM](#).

Under the umbrella of **buildingSMART** there are a number of International Committees currently grappling with the problem of extending IFC's from Buildings to Bridges, an Alignment for Roads, and for Roads and Railways in general, and to move IFC's from uncoordinated flat space with (0,0) in the corner, to a GIS and 12d environment with world coordinates and projections.

Beware that some groups talk BIM but then insist on data being supplied in an unpublished, encrypted proprietary format. In Australia this is often Revit, Navisworks, DWG or DGN.

This is diametrically opposed to the International Open BIM concept but apart from pointing that out to your client, you are caught between a rock and a hard place.

However for those users caught in the encrypted proprietary format trap, we are working to help solve your problems. For an overview of where we are currently at, please see the information in the next section [11.2 The 12d Approach to BIM](#).

Continue to [11.2 The 12d Approach to BIM](#) or return to [11 BIM](#).

## 11.2 The 12d Approach to BIM

**12d Solutions** is actively involved in, and supporting, the International effort on IFC's.

**12d Solutions** is:

- (a) a founding member of the **Open BIM Alliance of Australia** which supports the transferring of BIM data using the ISO Standard for IFCs.
- (b) working with **buildingSMART** (the International Group looking after IFC's) on how non-Building data should be transferred using IFC's. See <http://www.buildingsmart-tech.org>.
- (c) working with **buildingSMART** on how IFCs can be extended to Alignments, Roads and Railways.
- (d) working on the **Precinct Information Model** project, part of the **CRC for Low Carbon Living**.

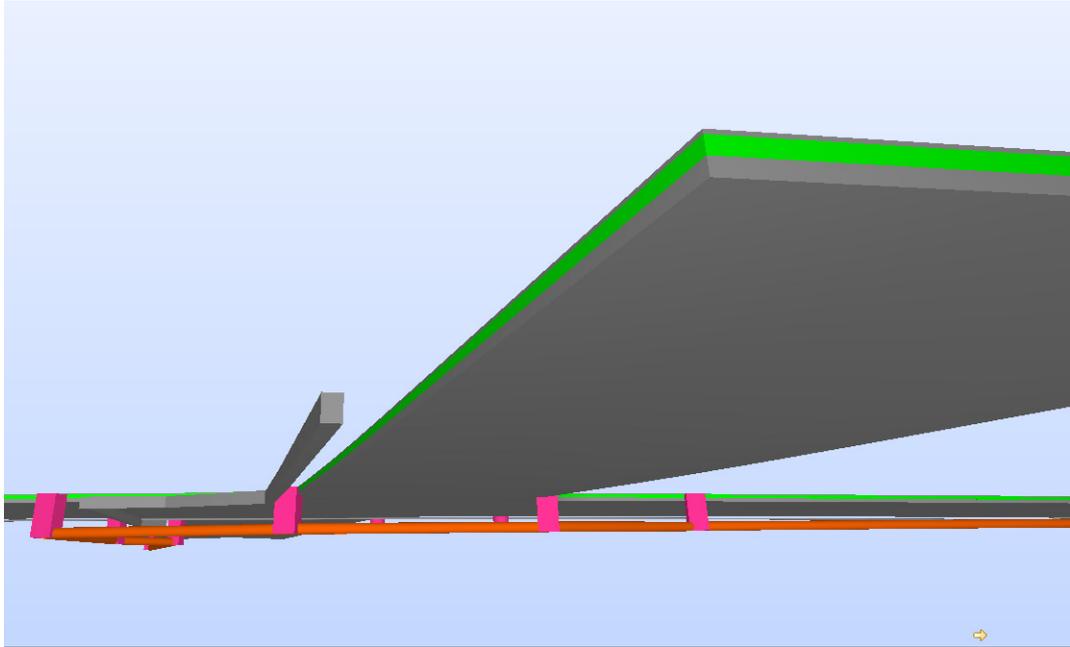
This project is looking at the problem of extending BIM from a few building on a site, to a larger area and including roads, drainage, sewerage etc.

- (e) developing methods of passing Civil objects such as road surfaces, pavement layers, kerb and gutters etc to BIM systems even though there are no currently standards for such data.

For example using existing IFC objects for writing out tins and trimeshes.



**Pavement Layers as Trimeshes in 12d Model**



Trimeshes from 12d Model loaded into the BIM Server Solibri using IFC's

For more information on **12d Model** and IFCs, see [15.2.7 IFC Output](#)

For those users caught in the encrypted proprietary format trap, although we do not have access to the formats, **12d Solutions** will continue to look for ways to make that data accessible to our users.

To this end, the approaches we are pursuing with our users to write out BIM type data from **12d Model** are:

**(a) Writing Out Fully Structured IFCs**

Using IFCs correctly means that the data to be output must have a fully defined hierarchical structure (known as the Spatial Structure).

This is the approach of the **12d Model IFC Writer**. See [15.2.7 IFC Output](#).

**(b) Writing Out Unstructured IFCs**

The IFC format could also be used in an informal way as a method of transferring data where there is no concern for the spatial structure but just after the object shapes.

12d is currently working on an alternate IFC Writer to do this.

**(c) Writing out OBJ and STL Files**

Drainage and sewer strings, super strings with round or rectangular sections, extrusions and trimeshes can be written out as OBJ files (see [15.2.8 Export OBJ](#)) or STL files ([15.2.9 Export STL](#)).

**(d) Writing out Polyface Mesh in DWG**

Drainage and sewer strings, super strings with round or rectangular sections, extrusions and trimeshes can be written out to DWG files as Polyface Meshes. See [15.2.4 Trimesh to DWG](#).

**(e) Revit Routines to Read in 12d Model Data.**

EXDS have developed routines that run from within Revit to read in **12d Model** data in 12d.XML format See [www.EXDS.com.au](http://www.EXDS.com.au).

The approaches we are using to bring data into **12d Model** are:

**(a) Reading in OBJs**

OBJ files can be read into to 12d Model as trimeshes. See [15.1.7 Wavefront OBJ Input](#)

**(b) Reading in Polyface Mesh in DWG**

Polyface mesh in a DWG file can be read into to 12d Model as trimeshes. See [15.1.5 DWG PolyFace Mesh to Trimesh](#)

**(c) Getting Revit Data into 12d Model**

Although the Revit database itself is unreadable, Revit can write data out in the Autodesk fbx format and there is an option in **12d Model** to read in an fbx file trimeshes. See [9.1.11 FBX Input](#).

Also if data can be written out from Revit as Polyface Meshes to a DWG file then again that can be read in into **12d Model** as trimeshes. See [15.1.5 DWG PolyFace Mesh to Trimesh](#).

Continue to [11.3 The UK Government and BIM](#) or return to [11 BIM](#).

## 11.3 The UK Government and BIM

In 2011, the UK Government released the **Government Construction Strategy** which was the framework for a range of work streams, all of which have the ultimate aim of reducing the cost of government construction projects by 15-20% (<https://www.gov.uk/government/publications/government-construction-strategy>).

One problem was identified in Section 2.30:

**2.30** *A lack of compatible systems, standards and protocols, and the differing requirements of clients and lead designers, have inhibited widespread adoption of a technology which has the capacity to ensure that all team members are working from the same data, and that:*

- s *the implications of alternative design proposals can be evaluated with comparative ease;*
- s *projects are modelled in three dimensions (eliminating coordination errors and subsequent expensive change);*
- s *design data can be fed direct to machine tools, creating a link between design and manufacture and eliminating unnecessary intermediaries; and*
- s *there is a proper basis for asset management subsequent to construction.*

and the remedy, BIM, was mandated in Section 2.31:

**2.31** *The Cabinet Office will co-ordinate Government's drive to the development of standards enabling all members of the supply chain to work collaboratively through Building Information Modelling (BIM). This will be a phased process working closely with industry groups, in order to allow time for industry to prepare for the development of new standards and for training*

The **Strategy** also included an **Action Plan** with a timetable, and Section 7 was for Building Information Modelling (BIM):

**7(i)Objective:** *To introduce a progressive programme of mandated use of fully collaborative Building Information Modelling for Government projects by 2016.*

From the **Government Construction Strategy: One Year On and Action Plan Update** it reported for BIM:

*A reciprocal Memorandum of Agreement has been reached with buildingSMART US to develop a national BIM standard with the principles of interoperability. This will be key to future delivery of Level 3 "Open & Shared" BIM.*

Continue to [11.4 Common Terms in BIM](#) or return to [11 BIM](#).

## 11.4 Common Terms in BIM

Here is a list of some need-to-know BIM terms and their definitions taken from the **NBS** website <http://www.thenbs.com>.

**NBS** is part of [RIBA Enterprises Ltd](#), which is wholly owned by the [Royal Institute of British Architects \(RIBA\)](#) and was contracted by the UK Government to build the National BIM Library.

### 1. . 4D, 5D, 6D

First there was 2D CAD, then 3D CAD – now there are extra dimensions to refer to the linking of the BIM model with time-, cost- and schedule-related information (although the precise order hasn't to date been agreed across the whole industry).

### 2. . Asset Information Model (AIM), Building Information Model (BIM)

Not only is there the 'Building' information model, but the 'Asset' information model – which is the name given to the same model post-construction, i.e. supplemented with the data needed to assist in the running of the completed asset. Note that 'asset' can also refer to civil engineering and infrastructure work.

### 3. . Common Data Environment (CDE)

This is a central information repository that can be accessed by all stakeholders in a project. Whilst all the data within the CDE can be accessed freely, ownership is still retained by the originator.

The scope and requirements for a CDE are defined in PAS 1192-2 (see [8. . PAS 1192](#)).

### 4. Level 0 BIM, Level 1 BIM, Level 2 BIM, Level 3 BIM

The move to 'full' collaborative working via distinct and recognisable milestones, in the form of 'levels'. These have been defined within a range from 0 to 3, and, whilst there is some debate about the exact meaning of each level, the broad concept is:

**Level 0** – no collaboration.

2D CAD drafting only. Output and distribution is via paper or electronic prints, or a mixture of both.

**Level 1** – a mixture of 3D CAD for concept work, and 2D for drafting of statutory approval documentation and Production Information.

CAD standards are managed to BS 1192:2007, and electronic sharing of data is carried out from a common data environment (CDE), often managed by the contractor.

There is no collaboration between different disciplines – each publishes and maintains its own data.

**Level 2** – collaborative working – all parties use their own 3D CAD models.

Design information is shared through a common file format, which enables any organisation to be able to combine that data with their own in order to carry out interrogative checks on it.

Hence any CAD software that each party used must be capable of exporting to a common file format.

This is the method of working that has been set as a minimum target by the UK government for all public-sector work, by 2016.

**Level 3** – integrated working between all disciplines by using a single, shared project model which is held in a common data environment.

All parties can access and modify that same model, removing the final layer of risk for conflicting information. This is known as 'Open BIM' (see [7. Open BIM](#)), and the UK government's target date for public-sector working is 2018, although the precise requirements have yet to be determined.

Note that the definition of BIM maturity Level 2 was originally developed as part of the UK Government strategy in 2011. It is also defined in PAS 1192-2, with reference to best practice and the adoption tools and standards.

It is also worth noting, though, PAS 1192-2 acknowledges that, given the early stages of adoption of managed methods of working in BIM at the time the PAS was drafted, it can be expected that Level 2 practices will continue to evolve, and that the scope of information sharing and exchange will vary from project to project.

Therefore, PAS 1192-2 anticipates that the definition of Level 2 BIM will continue to evolve around the core principles of the shared use of individually authored models in a CDE.

## 5. Construction Operations Building Information Exchange (COBie)

**COBie** is a data schema which is delivered in a spreadsheet data format, and contains a 'subset' of the information in the building model (all except graphical data, and hence a subset of IFC (see [6. Industry Foundation Class \(IFC\)](#)), for FM handover. It was originally devised by the US Army Engineering Corps.

Over the course of a project, data can be added to it from a range of sources (besides CAD programs), relating to brief, design, construction, operation, refurbishment or demolition, as the case may be.

The UK Government's Level 2 - mandated requirement is for COBie-compliant information exchange. BS 1192-4 documents best practice for the implementation of COBie.

## 6. Industry Foundation Class (IFC)

IFC is an object-based format, to enable exchange of information between different software. Developed by **buildingSMART**, a global alliance specialising in open standards for BIM, IFC is an official standard, BS ISO 16739, and contains geometric as well as other data.

## 7. Open BIM

An open-source approach to collaborative design, realisation and operation of buildings, based on open standards and work flows.

Open BIM is an initiative of several leading software vendors using the buildingSMART Data Model, which incorporates data to ISO 16739 (via the IFC file format), terms to ISO 12006-3 (using the International Framework for Dictionaries, which maps different technical terms that have the same meaning) and process to ISO 29481-1 (the Information Delivery Manual).

## 8. . PAS 1192

The PAS 1192 framework sets out the requirements for the level of model detail (the graphical content), model information (non-graphical content, such as specification data), model definition (its meaning) and model information exchanges:

- (a) PAS 1192-2 deals with the construction (CAPEX) phase, and specifies the requirements for Level 2 maturity; sets out the framework, roles & responsibilities for collaborative BIM working; builds on the existing standard of BS 1192, and expands the scope of the Common Data Environment (see [3. . Common Data Environment \(CDE\)](#)).
- (b) PAS 1192-3 deals with the operational (OPEX) phase, focussing on use & maintenance of the Asset Information Model, for Facilities Management.
- (c) BS 1192-4 documents best practice for the implementation of COBie.

(d) PAS 1192-5 is currently under development, and will cover security of data.

Continue to [11.5 What is Wrong with Local Coordinates ?](#) or return to [11 BIM](#).

## 11.5 What is Wrong with Local Coordinates ?

What we mean by Local Coordinates is the placing of (0,0) somewhere on the site, with a direction for the x or y axis, NOT a cartographic projection with (0,0) defined on the site.

There is nothing wrong with using a Local Coordinate system as long as you are aware of the restrictions and consequences. As Einstein said for General Relativity "space is locally flat".

So as long as you are working on a small site (say under 5 kilometres long) and everyone is able to reliably measure back to your (0,0) and use your definition of angle of the coordinate axes, then there should be no major problems.

But if you are working on a large project, or trying to combine data from a variety of sources, or want to coordinate work with a GPS, then you need to know more about how and why there are problems with Local Coordinates.

One of the problems is that the earth is not flat.

See

[11.5.1 An Exact Position for Everything on the Earth](#)

[11.5.2 Map \(Cartographic\) Projections - Eastings and Northings](#)

[11.5.3 Why Isn't MGA Used on a Small Building Site ?](#)

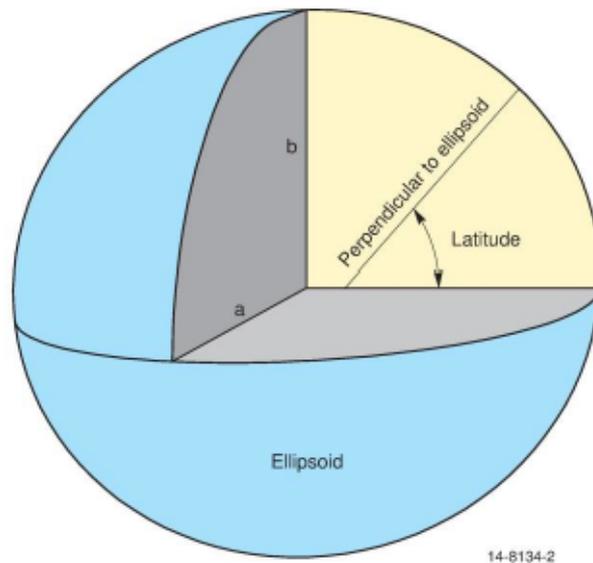
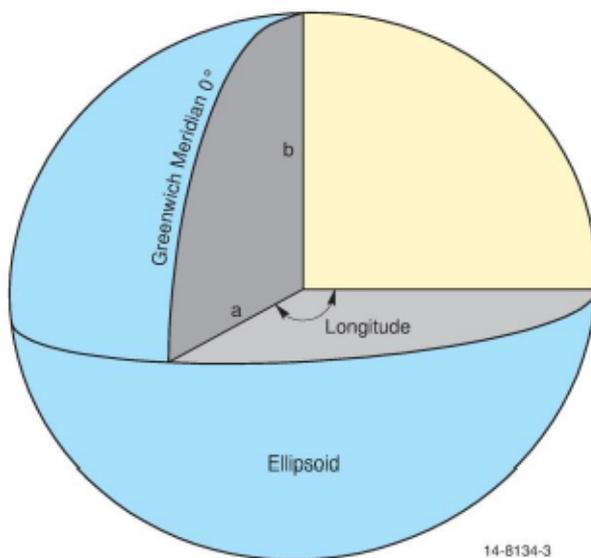
## 11.5.1 An Exact Position for Everything on the Earth

To give everything on earth a unique coordinates that fit in with GNSS systems (GPS, GLONASS, BeiDou and Galileo), a **reference ellipsoid** has been defined with:

- (a) (0,0,0) at the Mass centre of the earth (geocentre), a semi major axis of 6,378,137 m, and an inverse flattening of 298.257222101. This is known as **GRS80**.

This is because satellites orbit around the mass centre of the earth.

- (b) Longitude is an angular quantity measured from the Greenwich meridian.
- (c) Latitude is an angular quantity measured from the equatorial plane, to the plane defined by the point position and the plumb line to the ellipsoid surface.
- (d) Ellipsoid height is the height above the reference ellipsoid.



Using a GNSS system, the Geodetic coordinates of (Latitude, Longitude, Ellipsoid height) can be directly obtained for any position on earth that can see the satellites.

**Note:** Heights in Australia are not usually quoted as Ellipsoid heights  $h$  but instead in AHD (Australian Height Datum)  $H$  which is defined in terms of Means Sea Level. The difference between the ellipsoid height and the AHD is known as the **N (AHD Value)**.

$$\text{AHD (Geoid height)} = \text{Ellipsoid height} - \text{N (AHD) Value}$$

Continue to [11.5.2 Map \(Cartographic\) Projections - Eastings and Northings](#) or return to [11.5 What is Wrong with Local Coordinates ?](#) or [11 BIM](#).

## 11.5.2 Map (Cartographic) Projections - Eastings and Northings

Although Geodetic coordinate of Latitude and Longitude give unique coordinates for a point, they are not very appropriate for drawing up civil engineering projects. They are usually represented on a map by mathematically "projecting" them on to a surface, which can be laid flat.

A Map or Cartographic Projection is a transformation of the latitudes and longitudes of locations on the surface of an ellipsoid into locations on a plane. The projection coordinates are usually referred to as Eastings (x) and Northings (y).

The important thing is that the Cartographic projection can be reversed. That is, has an inverse. So the Cartographic projection

(a) maps a (Latitude, Longitude) to a (Easting, Northing)

and

(b) the inverse projection maps an (Easting, Northing) to a (Latitude, Longitude).

Consequently if you have the (Easting, Northing) then can calculate the equivalent (Latitude, Longitude) and vice versa.

The most commonly used Cartographic projection for civil works is the **Transverse Mercator** projection.

In mathematical speak, **Transverse Mercator** projections are **conformal** projections and preserve angles locally, implying that they map infinitesimal circles of constant size anywhere on the Earth to infinitesimal circles of **varying sizes** on the map. In contrast, mappings that are not conformal distort most such small circles into ellipses of distortion. An important consequence of conformality is that relative angles at each point of the map are correct, and the local scale (although varying throughout the map) **in every direction around any one point is constant**.

Transverse Mercator projections are commonly used for road projects by the Road Authorities in each State of Australia. They are also used for the Meridional Circuits in New Zealand.

The Transverse Mercator projection is also used as the basis of the **UTM** (Universal Transverse Mercator). The UTM is not a single map projection but divides the Earth into sixty zones, each a six-degree band of longitude, and uses a Transverse Mercator projection.

In Australia, **MGA** coordinates (Map Grid of Australia) are based on the UTM using the ellipsoid GRS80 previously defined.

So an MGA coordinate (Easting, Northing) has a unique (Latitude, Longitude).

And a reading from a GNSS device (commonly known as GPS) uniquely converts to a MGA coordinate, and conversely a MGA coordinate converts to (Latitude, Longitude) for use in a GPS without knowing anything about a local coordinate system.

Continue to [11.5.3 Why Isn't MGA Used on a Small Building Site ?](#) or return to [11.5 What is Wrong with Local Coordinates ?](#) or [11 BIM](#).

### 11.5.3 Why Isn't MGA Used on a Small Building Site ?

It is very easy to gloss over one very important point about Transverse Mercator projections as used in MGA.

Transverse Mercator projections are **conformal projections** and although they **preserve angles**, they do **not preserve distances**.

So a small circle will transform to a small circle but the radius will vary depending on where you are in an MGA zone.

The amount of the distortion is known as a **scale factor** and it varies from point to point. Over a small distance the scale factor is almost a constant but that constant value **is rarely 1**.

What that means in practise is if you draw everything up in MGA coordinates, the actual distance between two points in **NOT the standard square root distance between the two points**.

This means that on a plan in MGA coordinates, you can't simply use a ruler to measure the distance between two points. You need to allow for the scale factor. Over a small distance the scale factor will be a constant (for example 0.9996) but over a large distance you need to need to do the correct Geodetic calculations.

This also means that in a standard drafting system, the usual distance between two points that is used in conventional drafting dimensioning options, will not give the correct value.

For those working in a local coordinate project you may have noticed the scale factor effect when you have identified two points a good distance apart and you also have their MGA coordinates. If you measure between the points in your local coordinate system and then calculate the square root distance between the two MGA points using the actual MGA coordinates then you won't get the same number.

This is probably why most projects defined in local coordinates only give the MGA coordinates for **one point** and a **rotation of the axes**.

The result of all this is that if you only have a small building site then you can obtain fairly accurate MGA coordinates by applying a 2D Helmert transformation to the local coordinates. That is, a translation, a rotation of the axis and a **constant multiplication factor** (which could be the Point Scale Factor of the given MGA point).

However, as the size of the project increases, the larger errors will be in the MGA coordinates for points further away from the origin.

Similarly if you apply a translation, rotating and scale to convert data given in MGA coordinates (say from a GIS system or the State Road Authority) to local coordinates, then there will be errors the further away from the "best fitted" point.

#### **Note**

If you are not a surveyor you will probably not be aware that in **12d Model**, you can specify the Cartographic projection that the coordinates are in and can then obtain the (Latitude, Longitude) for selected points, measure between points and get the square-root distance and also the **ellipsoid distance** between the points.

Continue to [11.6 IFCs](#) or return to [11.5 What is Wrong with Local Coordinates ?](#) or [11 BIM](#).

## 11.6 IFCs

The **IFC Output** options write out **12d Model** data in the IFC STEP format where IFC stands for **Industry Foundation Classes**.

The IFC data model is intended to describe building and construction industry data, and to date, only covers buildings although there are committees currently looking at extending it to other infrastructure areas such as bridges, roads, railways.

It is a platform neutral, open file format specification that is not controlled by a single vendor or group of vendors.

It is an object-based file format with a data model developed by **buildingSMART** (formerly the International Alliance for Interoperability, IA) to facilitate interoperability in the architecture, engineering and construction (AEC) industry, and is a commonly used collaboration format in **Building Information Modeling** (BIM) based projects. See <http://www.buildingsmart-tech.org>.

The IFC model specification is open and available. It is registered by **ISO** and is an official International Standard ISO 16739:2013. See [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=51622](http://www.iso.org/iso/catalogue_detail.htm?csnumber=51622) and <http://www.buildingsmart-tech.org/specifications/ifc-releases/summary>.

It should be noted that from the ISO site, the following are outside the scope of ISO 16739:2013:

- exchange format definitions outside of the domain of construction and facility maintenance;
- project structure and component breakdown structures **outside of building engineering**;
- behavioural aspects of components and other information items.

Before trying to write out an IFC file, **12d Model** Users must be aware of how the IFC data model is structured so that the data to be written out is correct. And how this data is to be structured may vary from client to client wanting their data in an IFC file.

For a general overview of IFC's, see [http://en.wikipedia.org/wiki/Industry\\_Foundation\\_Classes](http://en.wikipedia.org/wiki/Industry_Foundation_Classes)

The IFC data model is very rich, as you would expect it to be to cover every aspect of a building, but we will only be using a small subset of its capabilities.

All the information and data is contained in **ifcProject** and there is only one of these.

All the data in a **12d Model** project can only go out to the one **ifcProject**.

We will mainly be using **ifcProduct** which represents occurrences in space such as physical building elements and spatial locations.

**ifcProduct** is the **base class** for all physical objects and is subdivided into:

(a) spatial items

Spatial items include **ifcSite**, **ifcBuilding**, **ifcBuildingStorey** and **ifcSpace**.

The spatial structure elements are linked together, and to the *IfcProject*, by using the objectified relationships *IfcRelContainedInSpatialStructure* and *IfcRelAggregates*.

(b) physical elements

Physical building elements include **ifcWall**, **ifcBeam**, **ifcDoor**, **ifcWindow**, **ifcStair** etc.

(c) structural analysis items

(d) other concepts.

Products may have associated materials, shape representations, and placements in space.

For more information, see:

[11.6.1 Spatial Structures](#)

[11.6.2 IFC Definitions](#)

[11.6.3 Representation of 12d Objects as IFCs](#)

## 11.6.1 Spatial Structures

Within the **ifcProject**, an overall Spatial Structure is defined which has a strict hierarchical structure.

At the top level is the **Project**.

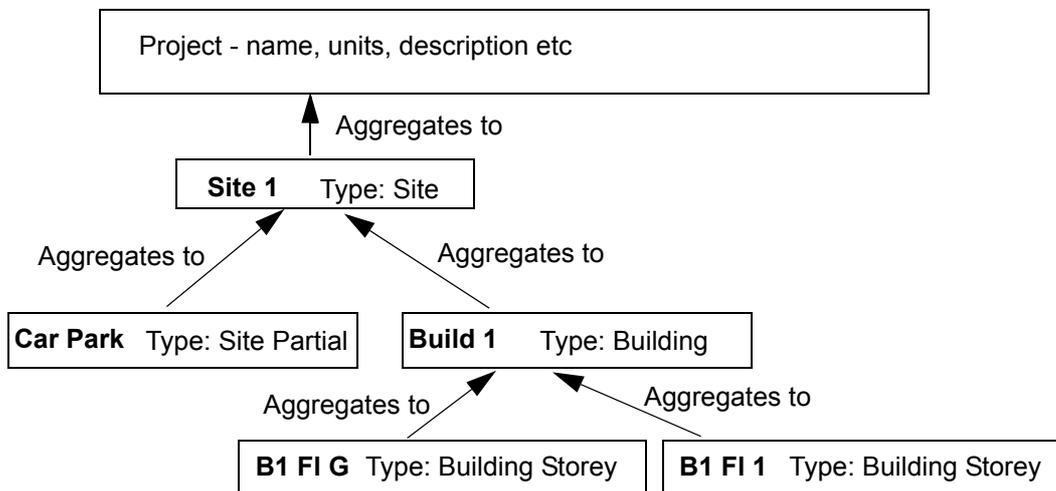
Then within this **Project** are **Sites**, and each **instance** of a **Site** has a **unique name**.

A Site can contain other Sites (called Site Partial) or Buildings, or Building Storeys or Spaces or Products.

A Building can contain Building Storeys, Spaces or Products.

Each instance of Site, Site Partial, Building, Building Storey, Space or Product has a unique name and is directly contained in only one other spatial structure.

The hierarchy of the overall Spatial Structure is specified by defining when each spatial structure **B** is directly contained in another spatial structure **A**. It is expressed as the spatial structure **B aggregates to** the spatial structure **A**.



So we only need to refer to a spatial structure by name and all the spatial structures contained beneath it are also uniquely specified.

In **12d Model** the Spatial Structure is constructed by the option

**File =>Data output =>IFC => Spatial structure**

which brings up the panel **IFC Spatial Structure**.

The spatial structures for the project, including the **Aggregates to**, are defined in the **Define** tab of the panel, and are stored as project attributes. See [15.2.7.4 Defining and Assigning Spatial Structures](#).

An IFC element is assigned to a spatial structure. Basic types of IFC elements are

- (a) building element
- (b) opening element
- (c) furnishing element
- (d) distribution element (including heating, ventilation, air conditioning, electrical and equipment elements)

(e) transportation element

**12d Model** data that is to be written out to an IFC file are written as **physical** elements and as such must belong to one spatial structure.

**12d Model** data is assigned a spatial structure by either defined a spatial structure for a whole model and/or for individual strings or trimeshes. The spatial structure for tins is taken to be that of the spatial structure of the model that the tin is in.

The **IFC Writer** panel writes out all the data of a user given spatial structure to a file. See [15.2.7.5 IFC Writer](#).

Continue to the next section [11.6.2 IFC Definitions](#) or return to [11.6 IFCs](#).

## 11.6.2 IFC Definitions

For some information on definitions of some of the IFC items used when writing the IFC file, see:

- [11.6.2.1 ifcProject](#)
- [11.6.2.2 ifcSite](#)
- [11.6.2.3 ifcElement](#)
- [11.6.2.4 ifcSpatialStructureElement](#)
- [11.6.2.5 ifcBuildingElement](#)
- [11.6.2.6 ifcBuildingElementProxy](#)
- [11.6.2.7 ifcFlowStorageDevice](#)
- [11.6.2.8 ifcFlowStorageDeviceType](#)
- [11.6.2.9 ifcFlowSegment](#)
- [11.6.2.10 ifcFlowSegmentType](#)
- [11.6.2.11 ifcRelContainedInSpatialStructure](#)
- [11.6.2.12 ifcRelAggregates](#)
- [11.6.2.13 ifcProduct](#)
- [11.6.2.14 ifcPropertySet](#)
- [11.6.2.15 IfcPropertySingleValue](#)
- [11.6.2.16 IfcShapeRepresentation](#)
- [11.6.2.17 IfcCircleHollowProfileDef](#)
- [11.6.2.18 IfcRectangleHollowProfileDef](#)

Do not be alarmed if the definitions need to be read a number of times to get your head around them. Remember they are only an extract of part of the definition and there are many more definitions that make up IFC's.

For the complete information on IFC's, see <http://www.buildingsmart-tech.org/specifications>

Or continue to the section [11.6.3 Representation of 12d Objects as IFCs](#) or return to [11.6 IFCs](#).

### 11.6.2.1 ifcProject

**Definition from IAI:** *The undertaking of some design, engineering, construction, or maintenance activities leading towards a product. The project establishes the context for information to be exchanged or shared, and it may represent a construction project but does not have to.*

All the information and data is contained in **ifcProject** and there is only one of these.

The **ifcProject** has information such as

- (a) the default units used

For setting the **ifcProject** units in **12d Model**, see [15.2.7.2 Project Units](#).

- (b) descriptive information about the project

For setting the **ifcProject** information in **12d Model**, see [15.2.7.3 Extra Project Details](#).

Within the **ifcProject**, a spatial structure is defined and it is a strict hierarchical structure. See [11.6.1 Spatial Structures](#).

Continue to [11.6.2.2 ifcSite](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.2 ifcSite

**Definition from ISO 6707-1:1989:** *Area where construction works are undertaken.*

**Definition from IAI:** *A defined area of land, possibly covered with water, on which the project construction is to be completed. A site may be used to erect building(s) or other AEC products.*

The geometrical placement of the site, defined by the *IfcLocalPlacement*, shall be always relative to the spatial structure element, in which this site is included, or absolute, i.e. to the world coordinate system, as established by the geometric representation context of the project. The world coordinate system, established at the *IfcProject.RepresentationContexts*, may include a definition of the true north within the XY plane of the world coordinate system.

An IFC project may span over several connected or disconnected sites. Therefore **site complex** provides for a collection of sites included in a project.

A site can also be decomposed in parts, where each part defines a **site section**.

Whether it is a site, site complex or site section is defined by the Composition Type attribute of the supertype *IfcSpatialStructureElement* which is interpreted as follow:

If the attribute value is **COMPLEX** then it is a **site complex**

If the attribute value is **ELEMENT** then it is a **site**

If the attribute value is **PARTIAL** then it is a **site section (site partial)**.

Note that in **12d Model**, you can currently only have a **site** and **site partial**.

Continue to [11.6.2.3 ifcElement](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.3 ifcElement

**Definition from IAI:** *Generalization of all components that make up an AEC product. Those elements can be logically contained by a spatial structure element that constitutes a certain level within a project structure hierarchy (e.g., site, building, storey or space). This is done by using the `IfcRelContainedInSpatialStructure` relationship.*

Elements are physically existent objects, although they might be void elements, such as holes.

EXAMPLEs of elements in a building construction context are walls, floors, windows and recesses.

There are various type of Elements derived from `ifcElement` including `ifcBuildingElement`, `ifcFurnishingElement`, `ifcElectricalElement` and `ifcBuildingElementProxy` (see [11.6.2.6 ifcBuildingElementProxy](#)).

Note that in **12d Model**, most data is written to the IFC file as `ifcBuildingElementProxy`.

Continue to [11.6.2.4 ifcSpatialStructureElement](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.4 ifcSpatialStructureElement

**Definition from IAI:** *A spatial structure element (`IfcSpatialStructureElement`) is the generalization of all spatial elements that might be used to define a spatial structure. That spatial structure is often used to provide a project structure to organize a building project.*

A spatial project structure might define as many levels of decomposition as necessary for the building project. Elements within the spatial project structure are:

- (a) site as `IfcSite`
  - (b) building as `IfcBuilding`
  - (c) storey as `IfcBuildingStorey`
  - (d) space as `IfcSpace`
- or
- (e) aggregations or parts thereof.

The Composition Type declares an element to be either an element itself, or an aggregation (complex) or a decomposition (part). The interpretation of these types is given at each subtype of `IfcSpatialStructureElement`. For example see [11.6.2.2 ifcSite](#).

The `IfcRelAggregates` is defined as an 1-to-many relationship and used to establish the relationship between exactly two levels within the spatial project structure.

Finally the highest level of the spatial structure is assigned to `IfcProject` using the `IfcRelAggregates`. See [11.6.2.12 ifcRelAggregates](#).

Continue to [11.6.2.5 ifcBuildingElement](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.5 ifcBuildingElement

**Definition from ISO 6707-1:1989:** *Major functional part of a building, examples are foundation, floor, roof, wall.*

**Definition from IAI:** *The building element comprises all elements that are primarily part of the construction of a building, i.e., its structural and space separating system.*

EXAMPLEs of building elements are walls, beams, or doors, they are all physically existent and tangible things.

Continue to [11.6.2.6 ifcBuildingElementProxy](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.6 ifcBuildingElementProxy

**Definition from IAI:** *The **IfcBuildingElementProxy** is a proxy definition that provides the same functionality as an **IfcBuildingElement**, but without having a defined meaning of the special type of building element, it represents.*

**NOTE1** The *IfcBuildingElementProxy* should be used to exchange special types of building elements for which the current IFC Release does not yet provide a semantic definition.

**NOTE2** The *IfcBuildingElementProxy* can also be used to represent building elements for which the participating applications can not provide additional semantic classification.

Continue to [11.6.2.7 ifcFlowStorageDevice](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.7 ifcFlowStorageDevice

The distribution flow element **IfcFlowStorageDevice** defines the occurrence of a device that participates in a distribution system and is used for temporary storage of a fluid such as a liquid or a gas (e.g., tank) or the voltage potential induced by the induced electron flow (such as a battery).

Its type is defined by **IfcFlowStorageDeviceType** or its subtypes. See [11.6.2.8 ifcFlowStorageDeviceType](#).

Continue to [11.6.2.8 ifcFlowStorageDeviceType](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.8 ifcFlowStorageDeviceType

The element type **IfcFlowStorageDeviceType** defines a list of commonly shared property set definitions of a flow storage device and an optional set of product representations. It is used to define a flow storage device specification (the specific product information that is common to all occurrences of that product type).

A flow storage device is a device used for the temporary storage of a fluid (such as a tank) or the voltage potential induced by the induced electron flow (such as a battery). Flow storage types (or the instantiable subtypes) may be exchanged without being already assigned to occurrences.

Continue to [11.6.2.9 ifcFlowSegment](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.9 ifcFlowSegment

The distribution flow element **IfcFlowSegment** defines the occurrence of a segment of a flow distribution system.

The **IfcFlowSegment** defines a particular occurrence of a segment inserted in the spatial context of a project. The parameters defining the type of the segment and/or its shape are defined by the **IfcFlowSegmentType**.

**IFC2x4 CHANGE** This entity has been deprecated for instantiation and will become ABSTRACT in a future release; new subtypes should now be used instead.

#### Material Use Definition

The material of the **IfcFlowSegment** is defined using one of the following entities:

**IfcMaterialProfileSetUsage:** for parametric segments, this defines the cross section and alignment to the 'Axis' representation, from which the 'Body' representation may be generated.

**IfcMaterialProfileSet:** for non-parametric segments (having fixed length or path), this may define the cross section for analysis purposes, however the 'Body' representation is independently generated.

**IfcMaterialConstituentSet:** for elements containing multiple materials where profiles are not applicable, this indicates materials at named parts.

**IfcMaterial:** for elements comprised of a single material where profiles are not applicable, this indicates the material.

The material is attached by the RelatingMaterial attribute on the IfcRelAssociatesMaterial relationship. It is accessible by the HasAssociations inverse attribute. Material information can also be given at the IfcFlowSegmentType, defining the common attribute data for all occurrences of the same type. Standard names and material types are defined at subtypes.

#### Geometry Use Definition

Standard representations are defined at the supertype **IfcDistributionFlowElement**. For parametric flow segments where IfcMaterialProfileSetUsage is defined and an 'Axis' representation is defined, then the 'Body' representation may be generated using the 'SweptSolid' or 'AdvancedSweptSolid' representation types by sweeping the profile(s) along the axis.

Continue to [11.6.2.10 ifcFlowSegmentType](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.10 ifcFlowSegmentType

The element type **IfcFlowSegmentType** defines a list of commonly shared property set definitions of a flow segment and an optional set of product representations.

It is used to define a flow segment specification (the specific product information, that is common to all occurrences of that product type).

A *flow segment type* is used to define the common properties of a flow segment that may be applied to many occurrences of that type. A flow segment is a section of a distribution system, such as a duct, pipe, or conduit, that typically has only two ports. Flow segment types (or the instantiable subtypes) may be exchanged without being already assigned to occurrences.

#### Material Use Definition

The material of the **IfcDistributionFlowSegmentType** is defined using one of the following entities:

**IfcMaterialProfileSet:** This defines the material cross section which may be used to generate the 'Body' representation at occurrences (for parametric definitions not having representation), or for analysis purposes.

**IfcMaterialConstituentSet:** For elements containing multiple materials where profiles are not applicable, this indicates materials at named aspects.

**IfcMaterial:** For elements comprised of a single material where profiles are not applicable, this indicates the material.

Continue to [11.6.2.11 ifcRelContainedInSpatialStructure](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.11 ifcRelContainedInSpatialStructure

**Definition from IAI:** *This objectified relationship, **ifcRelContainedInSpatialStructure**, is used to assign elements to a certain level of the spatial project structure. Any element can only be assigned once to a certain level of the spatial structure. The question, which level is relevant for which type of element, can only be answered within the context of a particular project and might vary within the various regions.*

EXAMPLE A multi-storey space is contained (or belongs to) the building storey at which its ground level is, but it is referenced by all the other building storeys, in which it spans. A lift shaft might be contained by the basement, but referenced by all storeys, through which it spans.

The **containment relationship** of an element within a spatial structure has to be a hierarchical relationship, **an element can only be contained within a single spatial structure element**.

The **reference relationship** between an element and the spatial structure may not be hierarchical, i.e. an element can reference many spatial structure elements.

**NOTE** The reference relationship is expressed by *ifcRelReferencedInSpatialStructure*.

Predefined spatial structure elements to which elements can be assigned are

- (a) site as *IfcSite* . See [11.6.2.2 ifcSite](#) .
- (b) building as *IfcBuilding*
- (c) storey as *IfcBuildingStorey*
- (d) space as *IfcSpace*

Continue to [11.6.2.12 ifcRelAggregates](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#) .

### 11.6.2.12 ifcRelAggregates

**Definition from IAI:** *The aggregation relationship **ifcRelAggregates** is a special type of the general composition/decomposition (or whole/part) relationship *ifcRelDecomposes*. The aggregation relationship can be applied to all subtypes of object.*

Some further specializations of decomposition may imply additional constraints and meanings, such as the requirement of aggregates to represent physical containment. In cases of physical containment the representation (within the same representation context) of the whole can be taken from the sum of the representations of the parts.

EXAMPLE: A roof is the aggregation of the roof elements, such as roof slabs, rafters, purlins, etc. Within the same representation context, e.g. the detailed geometric representation, the shape representation of the roof is given by the shape representation of its parts

Decompositions imply a dependency, i.e. the definition of the whole depends on the definition of the parts and the parts depend on the existence of the whole. The behaviour that is implied from the dependency has to be established inside the applications.

Continue to [11.6.2.13 ifcProduct](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#) .

### 11.6.2.13 ifcProduct

**Definition from IAI:** *Any object, or any aid to define, organize and annotate an object, that relates to a geometric or spatial context. Subtypes of `IfcProduct` usually hold a shape representation and a local placement within the project structure.*

This includes manufactured, supplied or created objects (referred to as **elements**) for incorporation into an AEC/FM project. This also includes objects that are created indirectly by other products, as spaces are defined by bounding elements. Products can be designated for permanent use or temporary use, an example for the latter is formwork. Products are defined by their properties and representations.

In addition to **physical products** (covered by the subtype [11.6.2.3 ifcElement](#)) and **spatial items** (covered by the subtype [11.6.2.4 ifcSpatialStructureElement](#)) the `IfcProduct` also includes non-physical items, that relate to a geometric or spatial contexts, such as grid, port, annotation, structural actions, etc.

Continue to [11.6.2.14 ifcPropertySet](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.14 ifcPropertySet

**Definition from IAI:** *The `IfcPropertySet` defines all dynamically extensible properties. The property set is a container class that holds properties within a property tree. These properties are interpreted according to their name attribute.*

Property sets, defining a particular type of object, can be assigned an object type (`IfcTypeObject`). Property sets are assigned to objects (`IfcObject`) through an objectified relationship (`IfcRelDefinedByProperties`). If the same set of properties applies to more than one object, it should be assigned by a single instance of `IfcRelDefinedByProperties` to a set of related objects. Those property sets are referred to as shared property sets.

#### Use Definition

Instances of `IfcPropertySet` are used to assign named sets of individual properties (complex or single properties). Each individual property has a significant name string. Some property sets have predefined instructions on assigning those significant name. The naming convention "Pset\_Xxx" applies to those property sets and shall be used as the value to the `Name` attribute.

In addition any user defined property set can be captured, those property sets shall have a `Name` value not including the **Pset\_** prefix.

Continue to [11.6.2.15 ifcPropertySingleValue](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.15 IfcPropertySingleValue

**Definition from IAI:** A property with a single value (`IfcPropertySingleValue`) defines a property object which has a single (numeric or descriptive) value assigned. It defines a property - single value combination for which the property name, the value with measure type (and optionally the unit) is given.

The unit is handled by the `Unit` attribute:

If the `Unit` attribute is not given, then the unit is already implied by the type of `IfcMeasureValue` or `IfcDerivedMeasureValue`. The associated unit can be found at the `IfcUnitAssignment` globally defined at the project level (`IfcProject.UnitsInContext`).

If the `Unit` attribute is given, then the unit assigned by the `Unit` attribute overrides the globally assigned unit.

Continue to [11.6.2.17 ifcCircleHollowProfileDef](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

### 11.6.2.16 IfcShapeRepresentation

**Definition from ISO/CD 10303-42:1992:** The shape representation is a specific kind of representation that represents a shape.

**Definition from IAI:** The *IfcShapeRepresentation* represents the concept of a particular geometric representation of a product or a product component within a specific geometric representation context.

The inherited attribute **RepresentationType** is used to define the geometric model used for the shape representation, the inherited attribute **RepresentationIdentifier** is used to denote the part of the representation captured by the *IfcShapeRepresentation* (e.g. Axis, Body, etc.).

Several representation types for shape representation are included as predefined types:

**Curve2D** 2 dimensional curves

**GeometricSet** points, curves, surfaces (2 or 3 dimensional)

**GeometricCurveSet** points, curves (2 or 3 dimensional)

**Annotation2D** points, curves (2 or 3 dimensional), hatches and text (2 dimensional)

**BoundingBox** simplistic 3D representation by a bounding box

**SectionedSpine** cross section based representation of a spine curve and planar cross sections. It can represent a surface or a solid and the interpolations of the between the cross sections is not defined

**MappedRepresentation** representation based on mapped item(s), referring to a representation map. Note: it can be seen as an inserted block reference. The shape representation of the mapped item has a representation type declaring the type of its representation items.

**SurfaceModel** face based and shell based surface model

**SolidModel** including swept solid, Boolean results and Brep bodies. More specific types are:

**SweptSolid** swept area solids, by extrusion and revolution

**Brep faceted** Brep's with and without voids

**CSG Boolean** results of operations between solid models, half spaces and Boolean results

**Clipping Boolean** differences between swept area solids, half spaces and Boolean results

**AdvancedSweptSolid** swept area solids created by sweeping a profile along a directrix

**NOTE** The definition of this entity relates to the STEP entity shape\_representation. Please refer to ISO/IS 10303-41:1994 for the final definition of the formal standard.

Continue to [11.6.2.17 IfcCircleHollowProfileDef](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

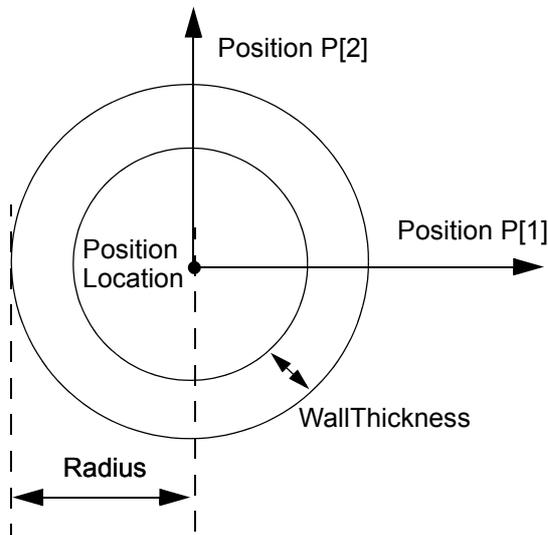
### 11.6.2.17 IfcCircleHollowProfileDef

**Definition from IAI:** The *IfcCircleHollowProfileDef* defines a section profile that provides the defining parameters of a circular hollow section (tube) to be used by the swept area solid.

Its parameters and orientation relative to the position coordinate system are according to the

following illustration.

The centre of the position coordinate system is in the profile's centre of the bounding box (for symmetric profiles identical with the centre of gravity).



**Position**

The parameterized profile defines its own position coordinate system. The underlying coordinate system is defined by the swept area solid that uses the profile definition. It is the xy plane of:

`IfcSweptAreaSolid.Position`

by using offsets of the position location, the parameterized profile can be positioned centric (using x,y offsets = 0), or at any position relative to the profile. Explicit coordinate offsets are used to define cardinal points (e.g. upper-left bound).

**Parameter**

The parameterized profile is defined by a set of parameter attributes, see attribute definition below.

**ifc Attribute definitions:**

**WallThickness:** Thickness of the material. It is the difference between the outer and inner radius.

The wall thickness must be smaller than the radius

Continue to [11.6.2.18 IfcRectangleHollowProfileDef](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

11.6.2.18 IfcRectangleHollowProfileDef

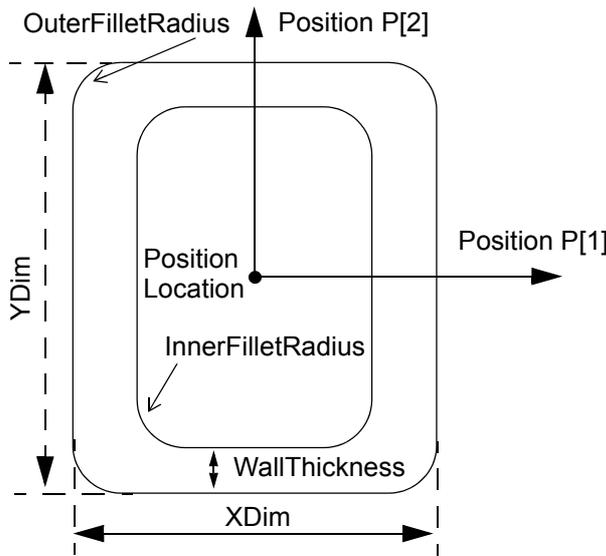
**Definition from IAI:** The *IfcRectangleHollowProfileDef* defines a section profile that provides the defining parameters of a rectangular (or square) hollow section to be used by the swept surface geometry or the swept area solid.

Its parameters and orientation relative to the position coordinate system are according to the following illustration.

A square hollow section can be defined by equal values for h and b.

The centre of the position coordinate system is in the profiles centre of the bounding box (for symmetric profiles identical with the centre of gravity).

Normally, the longer sides are parallel to the y-axis, the shorter sides parallel to the x-axis.



**Position**

The parameterized profile defines its own position coordinate system. The underlying coordinate system is defined by the swept area solid that uses the profile definition. It is the xy plane of:

`IfcSweptAreaSolid.Position`

by using offsets of the position location, the parameterized profile can be positioned centric (using x,y offsets = 0), or at any position relative to the profile. Explicit coordinate offsets are used to define cardinal points (e.g. upper-left bound).

**Parameter**

The parameterized profile is defined by a set of parameter attributes, see attribute definition below

**ifc Attribute definitions:**

**WallThickness:** Thickness of the material. The wall thickness shall be smaller than the X and Y dimension of the rectangle.

**InnerFilletRadius:** Radius of the circular arcs, by which all four corners of the outer contour of rectangle are equally rounded. If not given, zero (= no rounding arcs) applies.

The inner fillet radius (if given) shall be smaller than or equal to the X and Y dimension of the rectangle minus the wall thickness.

**OuterFilletRadius:** Radius of the circular arcs, by which all four corners of the outer contour of rectangle are equally rounded. If not given, zero (= no rounding arcs) applies.

The outer fillet radius (if given) shall be smaller than or equal to the X and Y dimension of the rectangle

Continue to the next section [11.6.3 Representation of 12d Objects as IFCs](#) or return to [11.6.2 IFC Definitions](#) or [11.6 IFCs](#).

## 11.6.3 Representation of 12d Objects as IFCs

Currently IFC's do not support many of the items in 12d Model so they are put out in ways that will fit in with the IFC schema.

See

[11.6.3.1 Super Alignment](#)

[11.6.3.2 Alignment](#)

[11.6.3.3 Drainage String](#)

[11.6.3.4 Sewer String](#)

[11.6.3.5 Tin](#)

[11.6.3.6 Trimesh](#)

### 11.6.3.1 Super Alignment

A **Super Alignment** is written out as an **IFCBUILDINGELEMENTPROXY** element.

The string attributes can be written out as an ifcPropertySet.

Continue to the [11.6.3.1 Super Alignment](#) or return to [11.6.3 Representation of 12d Objects as IFCs](#) or [11.6 IFCs](#).

### 11.6.3.2 Alignment

An **Alignment** is written out as an **IFCBUILDINGELEMENTPROXY** element.

The string attributes can be written out as an ifcPropertySet.

Continue to the [11.6.3.3 Drainage String](#) or return to [11.6.3 Representation of 12d Objects as IFCs](#) or [11.6 IFCs](#).

### 11.6.3.3 Drainage String

Each **Pit** is written out as an **IFCFLOWSTORAGEDEVICE** element.

For a circular pit, the shape representation is **ifcCircleHollowProfileDef**. See [11.6.2.17 IfcCircleHollowProfileDef](#).

For a rectangular pit, the shape representation is **ifcRectangleHollowProfileDef**. Note however that this has only one wall thickness so when writing the pit out, the outer shape is correct but the maximum value of the top, bottom, left and right thicknesses is used for the wall thickness. So if the top, bottom, left and right thickness are not all the same value then the inner rectangle is smaller than it actually is. See [11.6.2.18 IfcRectangleHollowProfileDef](#).

Each **Pipe** is written out as an **IFCFLOWSEGMENTS** element.

For a circular pipe, the shape representation is **ifcCircleHollowProfileDef**. [11.6.2.17 IfcCircleHollowProfileDef](#).

For a rectangular pipe, the shape representation is **ifcRectangleHollowProfileDef**. Note however that this has only one wall thickness so when writing the pipe out, the outer shape is correct but the maximum value of the top, bottom, left and right thicknesses is used for the wall thickness. So if the top, bottom, left and right thickness are not all the same value then the inner rectangle is smaller than it actually is.

Continue to the [11.6.3.4 Sewer String](#) or return to [11.6.3 Representation of 12d Objects as IFCs](#) or [11.6 IFCs](#).

### 11.6.3.4 Sewer String

Each **Maintenance hole** is written out as **IFCFLOWSTORAGEDEVICE** element.

For a circular maintenance hole, the shape representation is **ifcCircleHollowProfileDef**. [11.6.2.17 IfcCircleHollowProfileDef](#).

For a rectangular maintenance hole, the shape representation is **ifcRectangleHollowProfileDef**. Note however that this has only one wall thickness so when writing the maintenance hole out, the outer shape is correct but the maximum value of the top, bottom, left and right thicknesses is used for the wall thickness. This may mean that the inner rectangle may be smaller than it actually is.

Each **Pipe** is written out as an **IFCFLOWSEGMENT** element.

For a circular pipe, the shape representation is **ifcCircleHollowProfileDef**. [11.6.2.17 IfcCircleHollowProfileDef](#).

For a rectangular pipe, the shape representation is **ifcRectangleHollowProfileDef**. Note however that this has only one wall thickness so when writing the pipe out, the outer shape is correct but the maximum value of the top, bottom, left and right thicknesses is used for the wall thickness. So if the top, bottom, left and right thickness are not all the same value then the inner rectangle is smaller than it actually is.

Continue to the [11.6.3.5 Tin](#) or return to [11.6.3 Representation of 12d Objects as IFCs](#) or [11.6 IFCs](#).

### 11.6.3.5 Tin

A **Tin** is written out as an **IFCBUILDINGELEMENTPROXY** element.

The tin attributes can be written out as an ifcPropertySet.

Continue to the [11.6.3.6 Trimesh](#) or return to [11.6.3 Representation of 12d Objects as IFCs](#) or [11.6 IFCs](#).

### 11.6.3.6 Trimesh

A **Trimesh** is written out as an **IFCBUILDINGELEMENTPROXY** element.

The Trimesh attributes can be written out as an ifcPropertySet.

Return to [11.6.3 Representation of 12d Objects as IFCs](#) or [11.6 IFCs](#) or [11 BIM](#).

# 12. Bits and Pieces

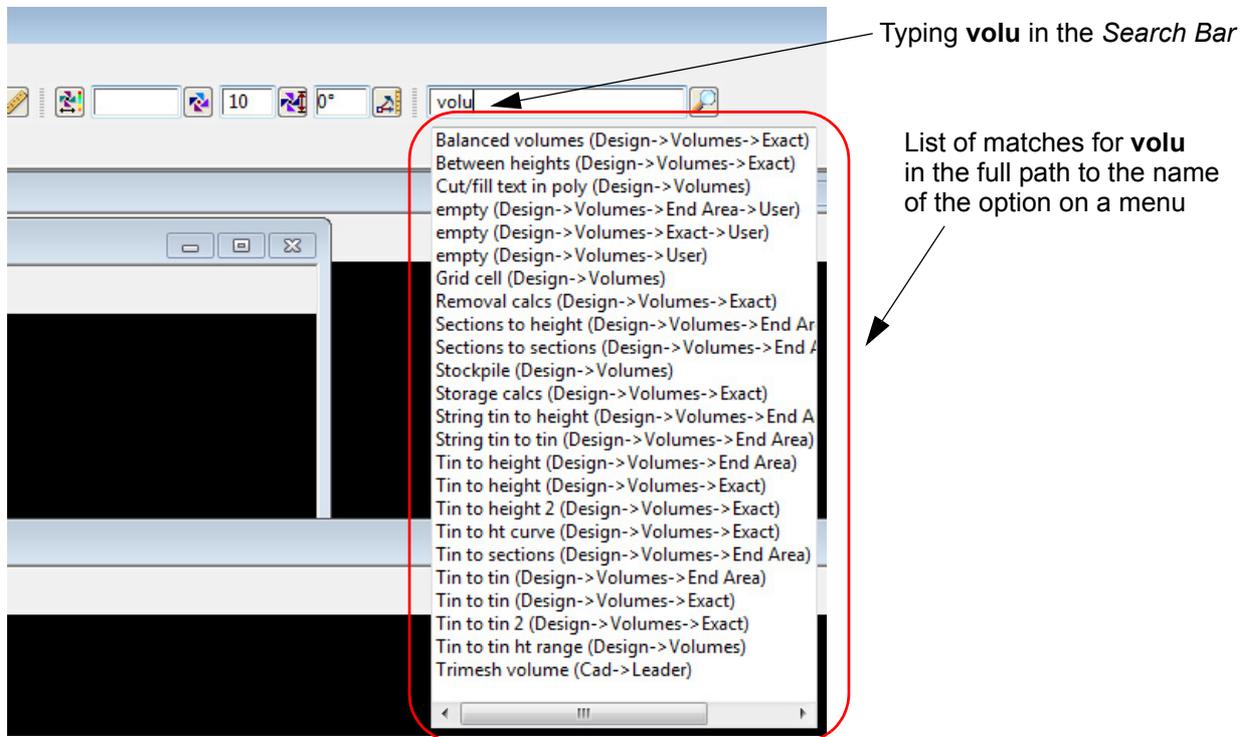
See

- [12.1 Search Looks at Full Menu Path](#)
- [12.2 Moving the Right Hand Side in Tree Controls](#)
- [12.3 Names.4d Comments in Data Tooltips](#)
- [12.4 New Plotter Type for PDF Plots](#)
- [12.5 Additions to Plotters.4d](#)
- [12.6 String/Model Name & Id Used in DXF, SLX Files](#)
- [12.7 String/Model Name & Id Used in Section View](#)
- [12.8 Demand Loading of Billboards & Textures](#)
- [12.9 MB in View Panel Field to Get View](#)
- [12.10 Resizing Panels](#)
- [12.11 Regions in Some Grids](#)
- [12.12 Extra Options on Model Icon Pop Up](#)
- [12.13 Extra Options on Tin Icon Pop Up](#)
- [12.14 Pipe ControlBar](#)
- [12.15 Attributes ControlBar](#)
- [12.16 Icons in Property Sheets](#)
- [12.17 More Borders Types for Text](#)
- [12.18 Arithmetic in an Integer Panel Field](#)
- [12.19 If Else in Real Value Fields](#)
- [12.20 Legacy Name in Colours.4d](#)
- [12.21 Colours in Pop Up Order](#)
- [12.22 Full Stop Allowed in Colour Name](#)
- [12.23 Subgroups in Linestyles.4d and Symbols.4d](#)

## 12.1 Search Looks at Full Menu Path

When text is typed into the **Search Bar**, the full path name of name of the option on the menu is now searched to find matches with the typed in text.

For example, typing in **volu** will bring up th list shown below.



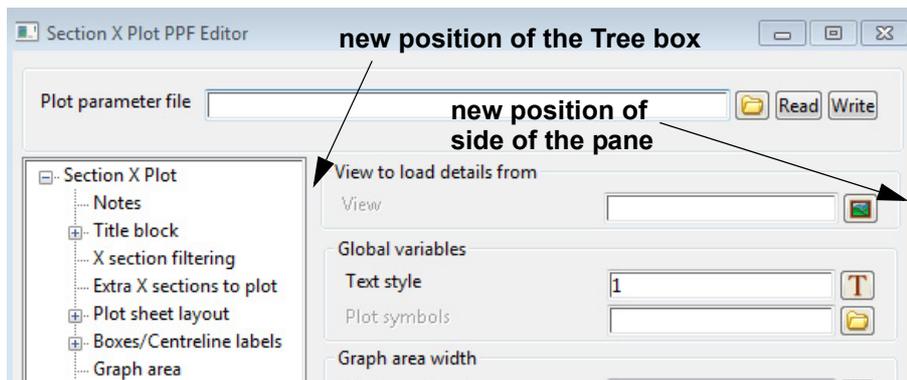
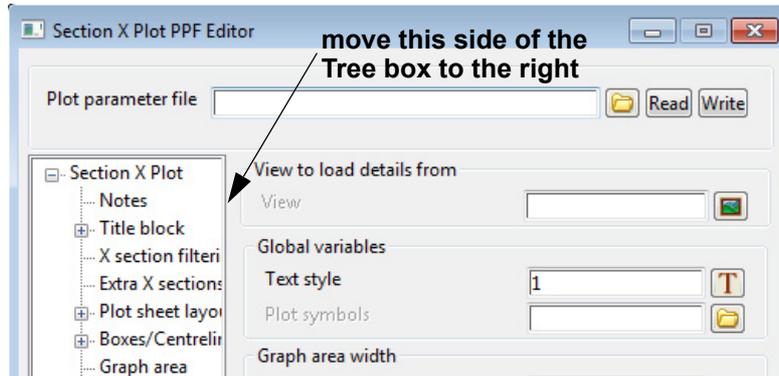
In V10, it would only have displayed **Balanced volumes**.

**Note** that case is ignored when searching for matches.

## 12.2 Moving the Right Hand Side in Tree Controls

In panels, moving the right hand side of a Tree control box to the right when the items on the right of the tree box have minimal widths, now also widens the panel so that you no longer have to first resize the panel before moving the Tree box side.

For example, in PPF Editors you often have to move the right hand side of the Tree box to the right to see some of node names.



The larger size of the panel is recorded when writing out **DDX** or **SLX** files.

### NOTE

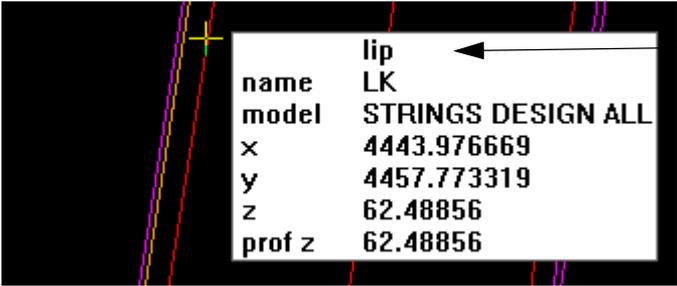
Moving the right hand side of the tree control box leaves the entire panel the same size. So you may need to resize the panel after doing so to make it smaller.

## 12.3 Names.4d Comments in Data Tooltips

When **Data tooltips** are turned on (**D** snap) and a string matches a row in the **Names.4d** file with a comment, then the comment is displayed at the top of the Data Tool tip.

part of **Names.4d** file

213	J*	optio	option:	cadast	green	line	1	optional	boundary join line
214	K*	optio	option:	roads	yellow	line	1	optional	top of kerb
215	L*	optio	option:	roads	yellow	line	1	optional	lip
216	N*	optio	option:	surv	red	line	1	option	ridge breaklines

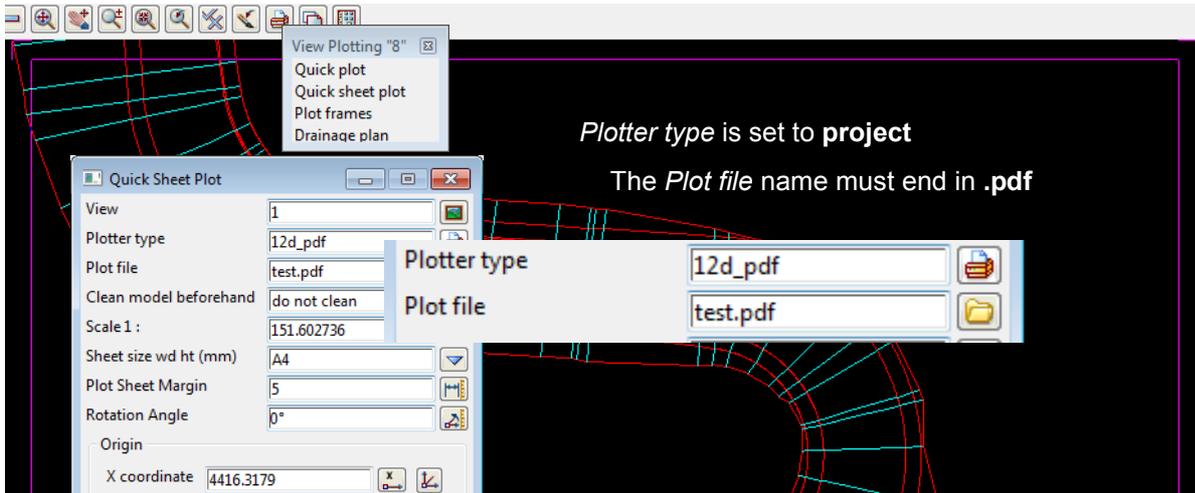
  


Comment taken from a match with L\* in the **Nmes.4d** file

## 12.4 New Plotter Type for PDF Plots

There is a new plotter type called **pdf\_12d** that replaces the V10 pdf plotter, *pdf\_writer*. It should now work for all plots.

After selecting **Plotter type** as **pdf\_12d**, fill in the name of the pdf file to be produced in the **Plot file** field. If there is no *.pdf* at the end of the file name then the option should automatically add it but this is not always working so for the moment, use a **Plot file** name that ends in **.pdf**.



### Important Note

The new plotter creates a temporary project in the working folder (or the folder where the pdf is being written to) as well as the pdf file. This should be automatically deleted after the pdf has been produced.

## 12.5 Additions to Plotters.4d

See

[12.5.1 pdf\\_12d Plotter Engine](#)

[12.5.2 string\\_weights Keyword](#)

[12.5.3 Group Keyword](#)

### 12.5.1 pdf\_12d Plotter Engine

There is a new **engine** called **pdf\_12d** in the *plotter definition* in the **plotters.4d** file.

For example

```
plotter "pdf_12d no string weights" {
  engine      pdf_12d
  extension   ".pdf"
  map_file    "lanes.pmf" // using plotter (pen) map file lanes.pmf
  colour      true
}
```

The **pdf\_12d** engine is to be used instead of the **engine pdf\_writer** that was in V10.

### 12.5.2 string\_weights Keyword

There is now an optional **string\_weights** keyword in the *plotter definition* in the **plotters.4d** file.

If **string\_weights = 0** or **false** then the plotter behaves as it does now (this is the default).

If **string\_weights = 1** or **true** then if the *weight* of the string is not zero then the **weight of the string** is used **instead** of any weight in the **plotter (pen) mapping file** (the weight in the *pmf* file).

If *string\_weights* is absent then the default value is **0 (false)**

For example, a pdf plotter that uses a pmf file called *lanes.pmf* but has string weights overriding the pmf entry for a pen/colour is

```
plotter "pdf_12d string weights" {
  engine      pdf_12d
  extension   ".pdf"
  map_file    "lanes.pmf" // using plotter (pen) map file lanes.pmf
  colour      true
  string_weights true
}
```

## 12.5.3 Group Keyword

There is now an optional **group** keyword in the *plotter definition* in the `plotters.4d` file so that the individual plotters given a group/subgroup structure.

If the **group** keyword for a *plotter definition* exists, then the information is used as a group/subgroup structure in any **Plotters** pop up.

The group/subgroup structure is written as the top group name first, followed by the first level subgroup name, the second level subgroup name, *etc.*, with each of the names separated by a forward slash `/`.

`group_name/first_level_subgroup_name/second_level_subgroup_name etc.`

For example, a top level group `Folder1` with the first level subgroup `Folder2` is written as

`Folder1/Folder2`

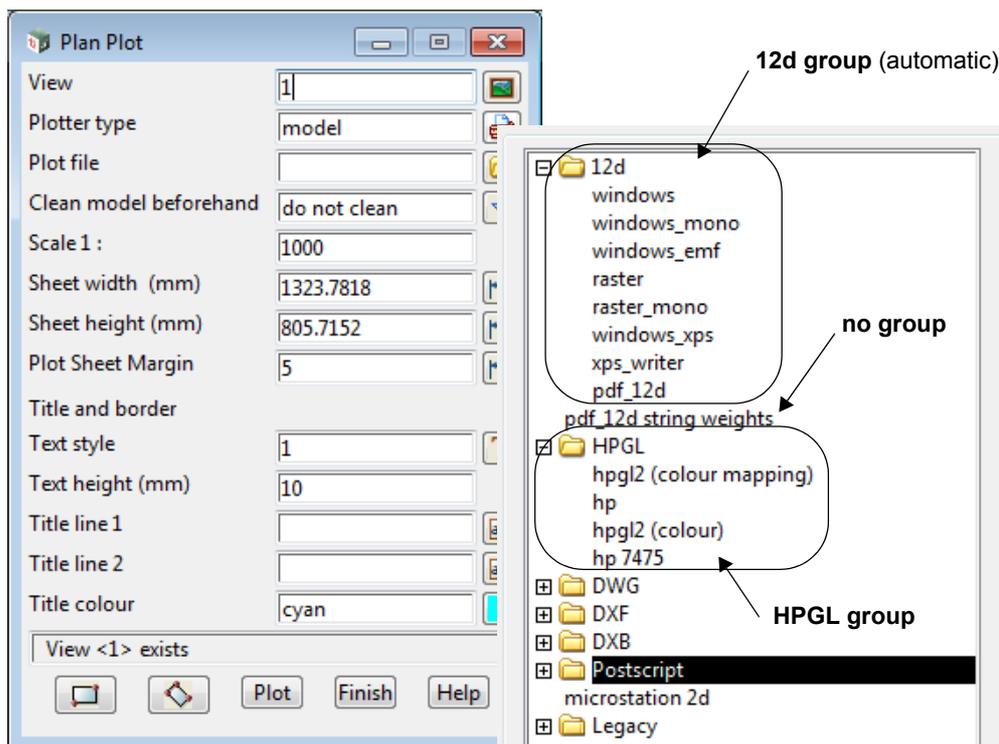
However, even though there are groups, the names of **each plotter** must be unique and it is the plotter name that is written to the Plotters box.

For example,

```
plotter "hpgl2 (colour mapping)" {
    // hpgl2 colour plotter
    // uses mapping file for colours

    engine hpgl2
    extension ".hpc"
    colour true
    map_pens true
    group "HPGL"
}
```

All the special **12d plotters** (`windows`, `windows_mono`, `pdf_12d` etc) are automatically placed in a group called **12d**.



## 12.6 String/Model Name & Id Used in DXF, SLX Files

With **String Select** boxes in panels, the **String name** and **String Id**, and the **Model name** and **Model Id** for the model the string is in, are used in a DDX or SLX files.

So when the SLX or DDX file is used, or an option recorded in a Chain is later run, if the String Id does not exist any more, then the string name and model id/name can be used to locate the string to use.

This means that options with **String Selects** that had the original strings deleted and recreated with the same name (for example in an **Apply MTF**), will now work. in Chains.

### Important Note

If the *String Id* no longer exists and the *String Name* is used to locate the string, then the first string found (in the model) with **matching String Name** will be used. So if there are many strings with the same name, there is no way to know which one will be used when restoring an SLX, a DDX or running a Chain.

## 12.7 String/Model Name & Id Used in Section View

For the string being profile on a Section View, the **String name** and **String Id**, and the **Model name** and **Model Id** for the model the string is in, are now recorded and used for the Section View. So if a string profiled on the Section View is deleted and recreated again with the same name, then the string name and model id/name can be used to locate a string to use for a reprofile.

If the string profiled on a Section View is deleted, when the cursor is next moved over the section view, the section view will search for the string using the string name and model Id/name, and if a string is found, the section view automatically reprofiles using the new string.

### Important Note

If the *String Id* no longer exists and the String Name used to locate a string, then the first string found (in the model) with a matching String Name will be used. So if there are many strings with the same name then there is no guarantee of which one will be used for reprofiling on the Section View.

## 12.8 Demand Loading of Billboards & Textures

The loading of **billboards.4d** and **textures.4d** has been changed so that they are demanded loaded rather than all loaded at once. That is, the individual billboards and textures are only loaded as they are needed.

This should significantly reduce the time spent loading billboards and textures, especially when only a small number of them are actually being used in a project.

## 12.9 MB in View Panel Field to Get View

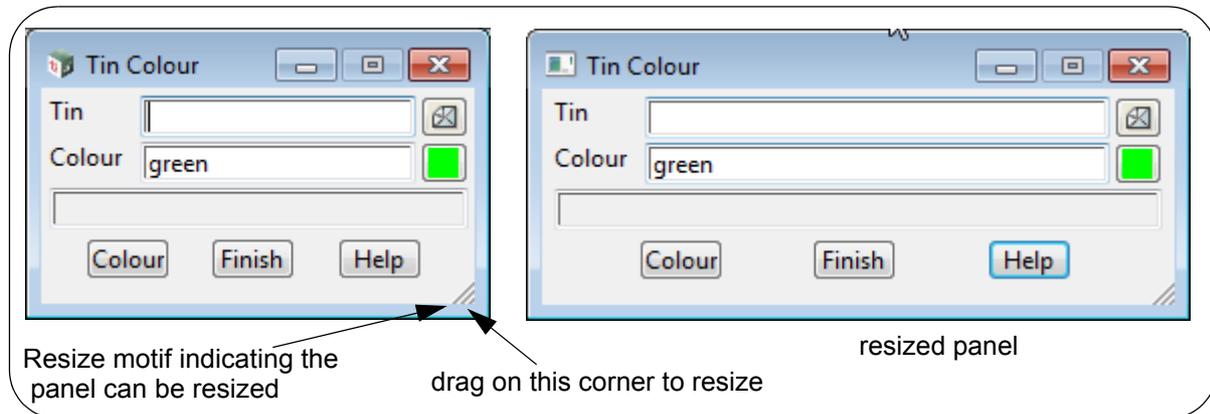
Clicking the middle mouse button (MB) in a **View** field will activate the **Same As** pick and if a string is then selected from a view, the name of the View that the string is on will be written to the **View** field.

Remember that when a view name is selected this way that <Enter> must be pressed for the **View** field to validate, and perform any other actions that the **View** field does if the view name was selected from the **View** icon popup.

## 12.10 Resizing Panels

Most panels now can now be resized.

If a panel can be resized then there is a Resize Motif (three lines) in the bottom right corner and dragging from that corner will resize the panel.



How the panel can be resized will depend on what items are in the panel. For example, the **Tin Colour** panel can only be made wider but cannot be made smaller or taller.

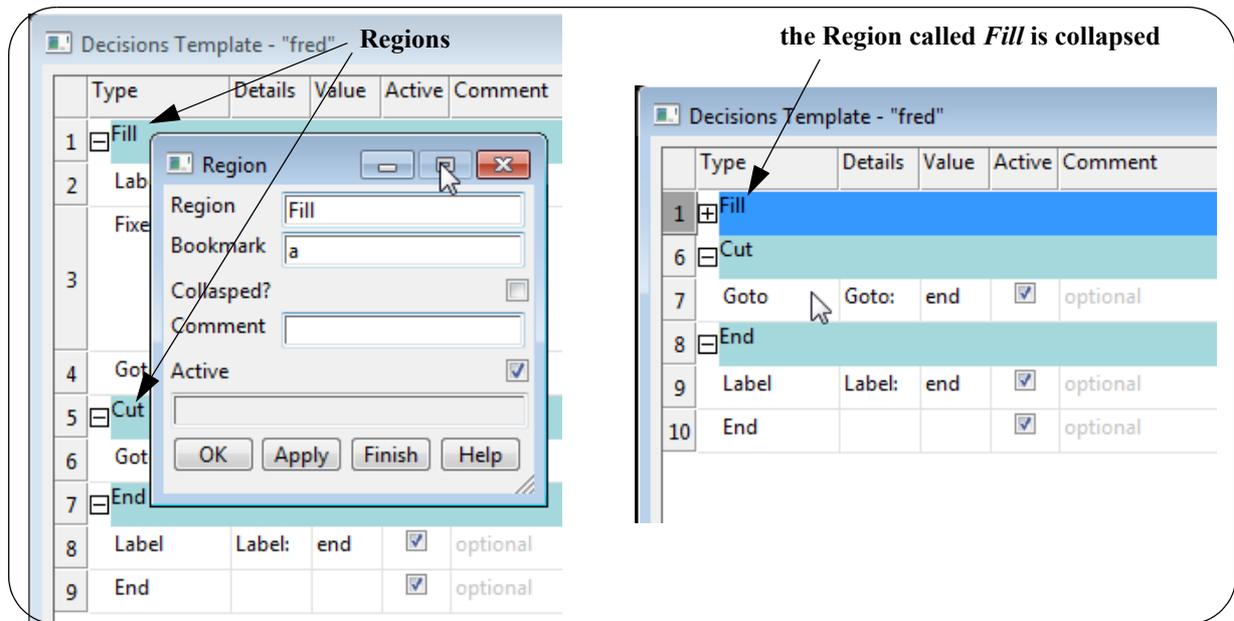
After a panel has been resized, the new size will be saved in a default file (.ddx), any screen layout file (.slx) and in any Chains.

## 12.11 Regions in Some Grids

In some grids there is a new **Region** command (rule).

A **Region** in a grid is a special command that has the property that a Region can be **collapsed**. And *collapsing* a Region means that all the rows after the Region command until the next Region command, are hidden in the grid. The row of the **Region** command is coloured light blue.

Regions must have a unique name within the grid.



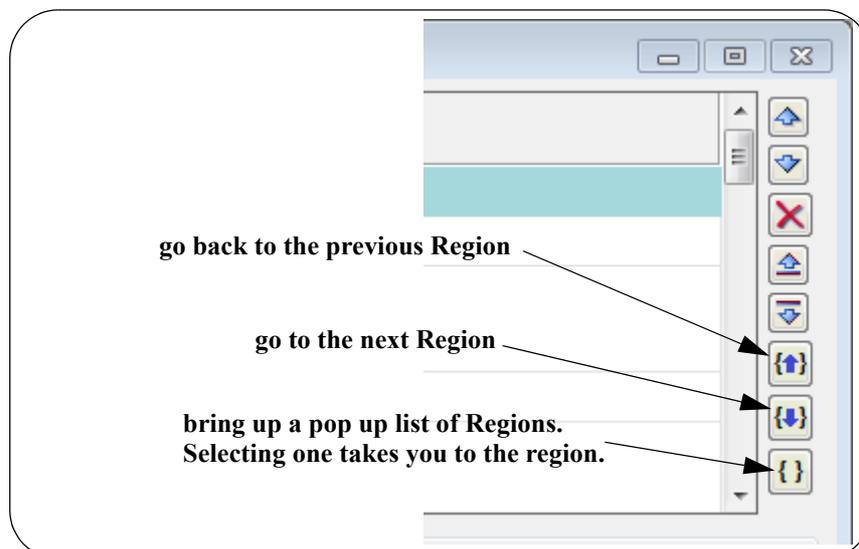
A **Region** can have a secondary name called a **Bookmark**. If no **Bookmark** is given then when the grid is saved, the **Bookmark** is given the same name as the Region.

At the right hand side of the grid, there are icons **Regions**, **Previous region** and **Next region**.

Clicking on the **Regions** icon will bring up a list of all the **Bookmarks** in the grid, and clicking on a **Bookmark** in the list will take you to that Region/Bookmark.

**Previous region** will take you from the highlighted row to the Region before the highlighted row.

**Next region** will take you from the highlighted row to the next Region after the highlighted row.



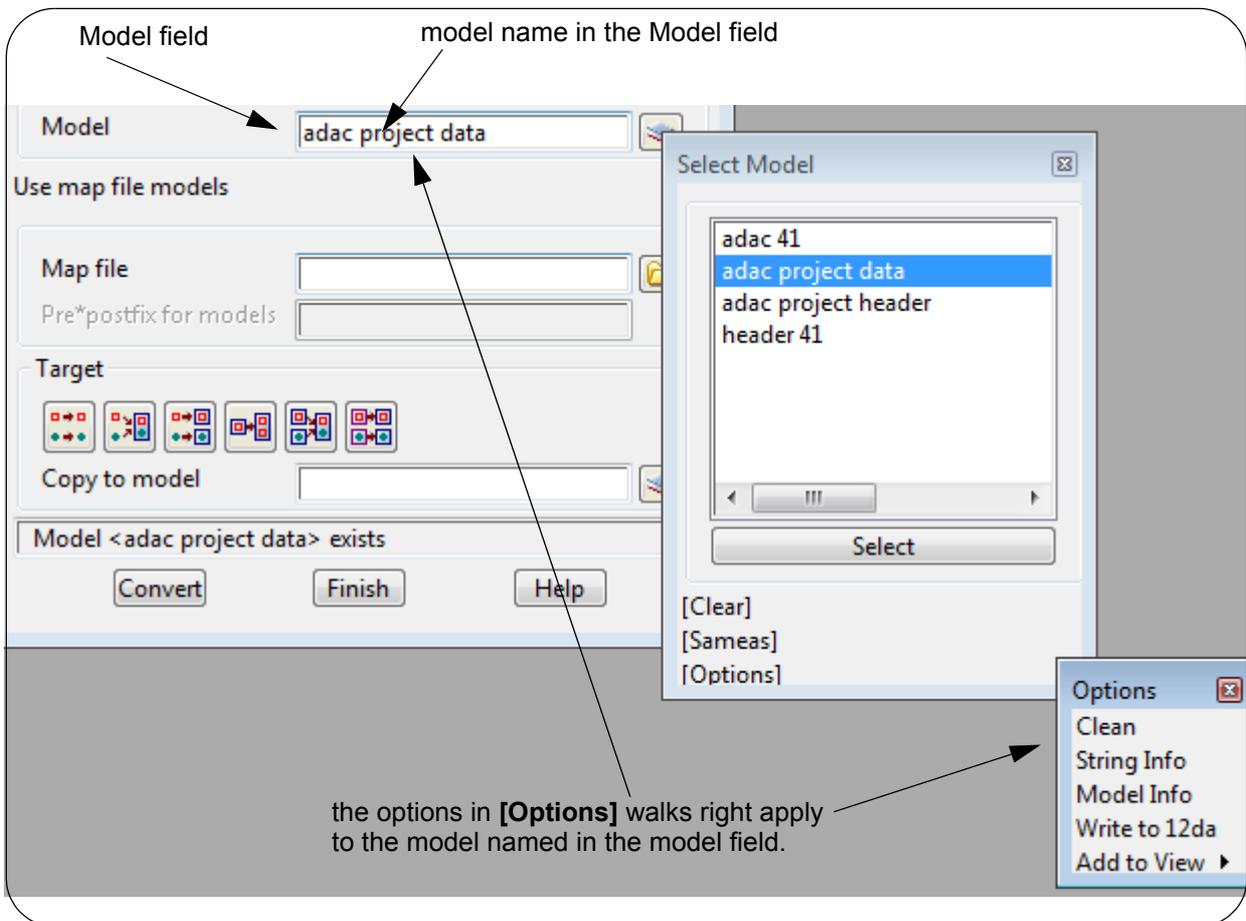
## 12.12 Extra Options on Model Icon Pop Up

When you have a **Model** field with a model name of an existing model in it, the **Select Model** pop up for the **Model** icon at the end of the **Model** field now has an **[Options]** at the bottom of the pop up with a walk right list of options that apply to the model in the **Model** field.

Note that the model must exist so if the **Model** field in a panel is one that creates the model, then using **[Options]** will only make sense **after** the panel has been run and the model has been created.

### Warning

The options on **[Options]** apply to the model in the **Model** field, **NOT** the model highlighted in the **Select Model** pop up.



**Clean** - cleans the model given in the **Model** field.

**String Info** - brings up the **String Information Table** panel and runs it for the model given in the **Model** field.

**Model Info** - brings up the **Model Information** panel and runs it for the model given in the **Model** field.

**Write to 12da** - brings up the **Write 12d Solutions Ascii Data** panel with the model given in the **Model** field set as the *Data Source*.

**Add to View** - adds the model given in the **Model** field to the view selected from the Add to View walk right list.

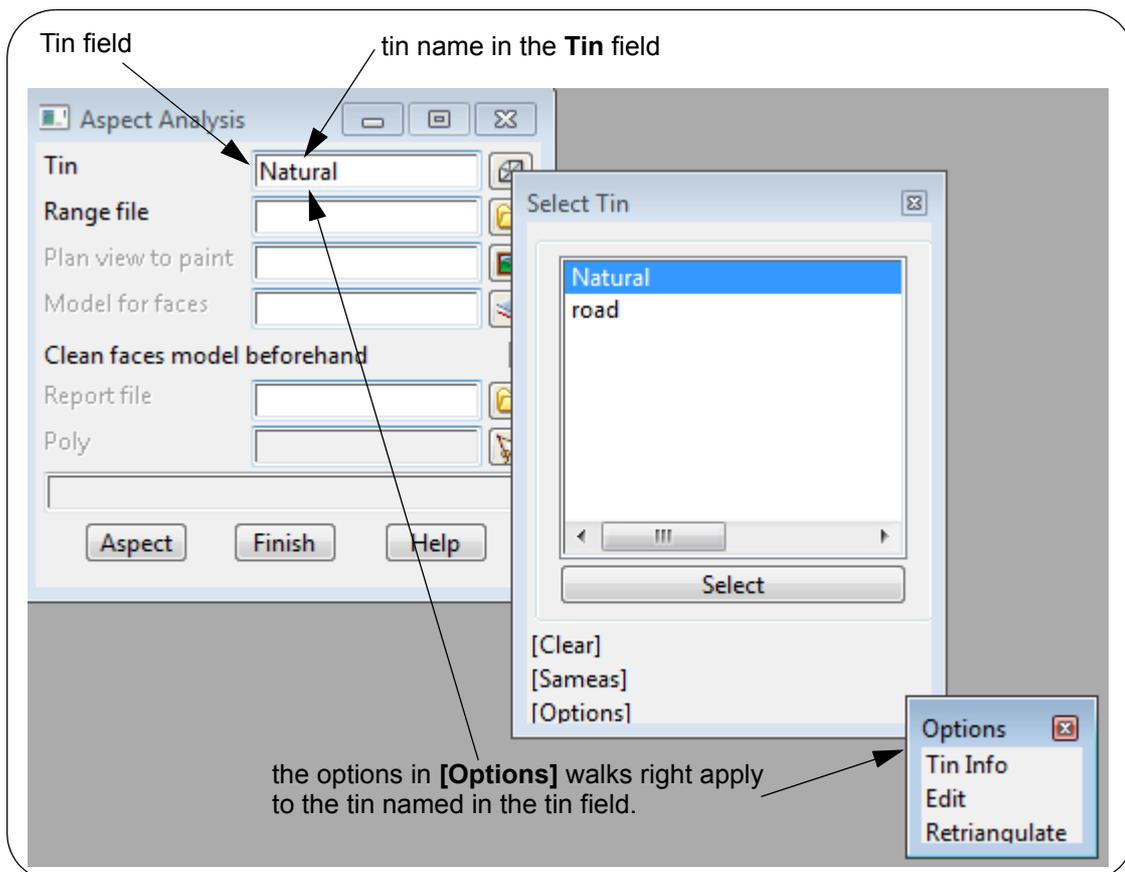
## 12.13 Extra Options on Tin Icon Pop Up

When you have a **Tin** field with a tin name of an existing tin in it, the **Select Tin** pop up for the **Tin** icon at the end of the **Tin** field now has an **[Options]** at the bottom of the pop up with a walk right list of options that apply to the tin in the **Tin** field.

Note that the tin must exist so if the Tin field in a panel is one that creates the tin, then using **[Options]** will only make sense **after** the panel has been run and the tin has been created.

### Warning

The options on **[Options]** apply to the tin in the **Tin** field, **NOT** the tin highlighted in the **Select Tin** pop up.



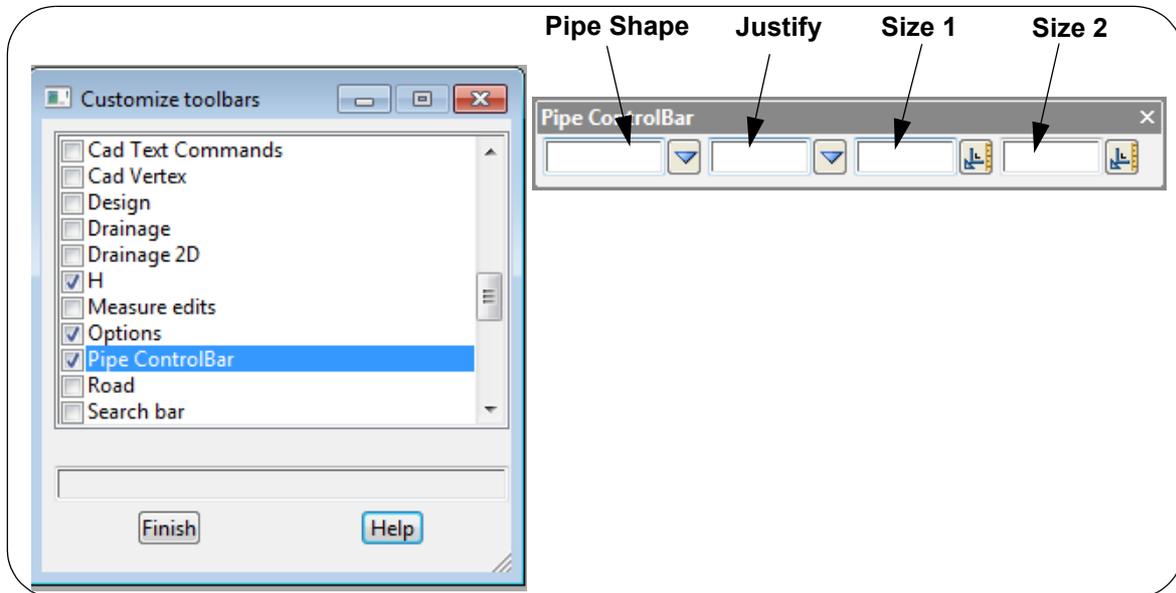
**Tin Info** - brings up the **Tin Information** panel and runs it for the tin given in the **Tin** field.

**Retriangulate** - runs retriangulate for the tin given in the **Tin** field.

**Edit** - brings up the **Retriangulate Tin** panel with the tin given in the **Tin** field set as the *Tin* and the information for that tin loaded into the panel.

## 12.14 Pipe ControlBar

There is a **Pipe ControlBar** and it is used to make a round or culvert string whenever a new string is created by the CAD options.



When a new string is created with the CAD options and the value in the **Shape** field of the **Pipe ControlBar** is **Diameter** or **Culvert**, then the **Justify**, **Size 1** and **Size 2** fields from the **Pipe ControlBar** are given to the created string.

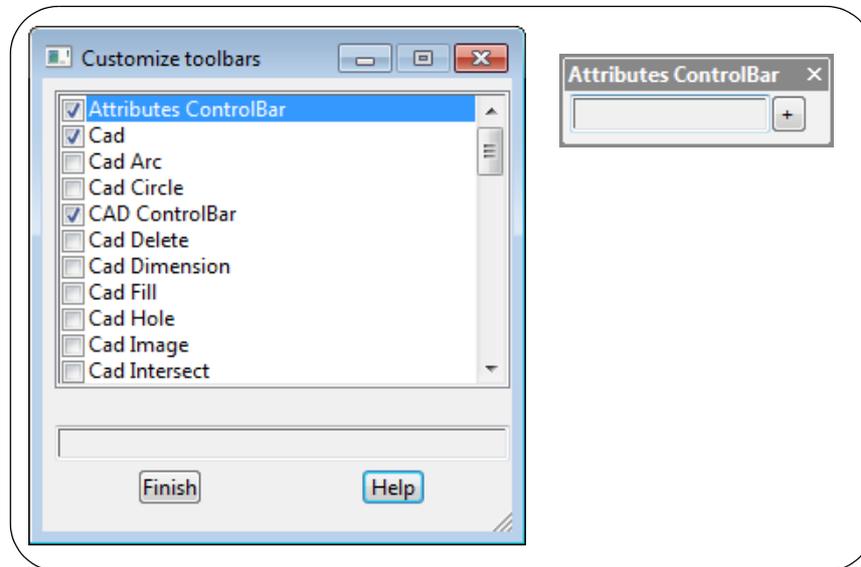
To clear **all** the fields in the Pipe ControlBar, clear the **Pipe Shape** field and press <Enter>. This clears all the other fields of the **Pipe ControlBar**.

### Note

The *Names.4d* now has a **Pipe** section so that the **Pipe ControlBar** can be automatically filled in when a name is selected in the **CAD ControlBar** that is in the **Pipe** section. See [4.4.3 Pipe Node in Names.4d](#).

## 12.15 Attributes ControlBar

There is a **Attributes ControlBar** and it is used to apply string attributes whenever a new super string is created by the CAD options, the attributes in the Attributes ControlBar are given to the string as string attributes.



When a new string is created with the CAD options, the string is given the attributes in the **Attributes ControlBar** as string attributes.

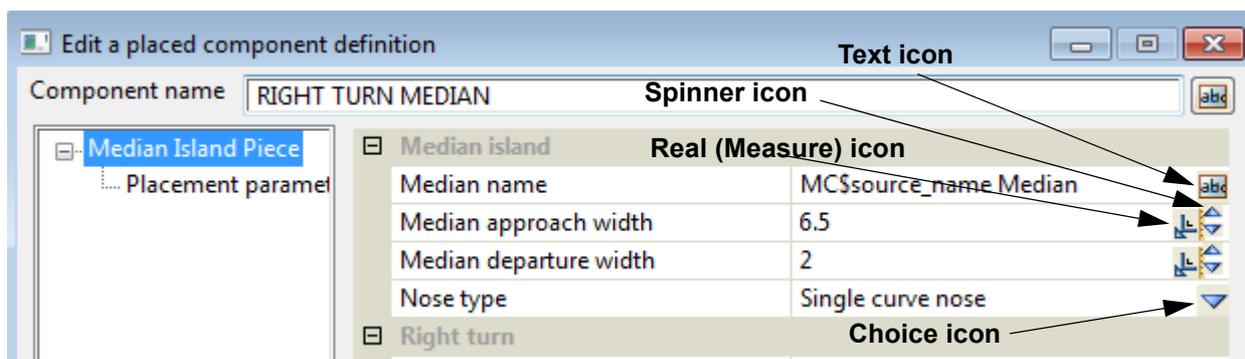
**Note**

The *Names.4d* now has an **Attribute Data** section so that the **Attributes ControlBar** can be automatically filled in when a name is selected in the **CAD ControlBar** that is in the **Attributes** section. See [4.4.4 Attributes Node in Names.4d](#).

## 12.16 Icons in Property Sheets

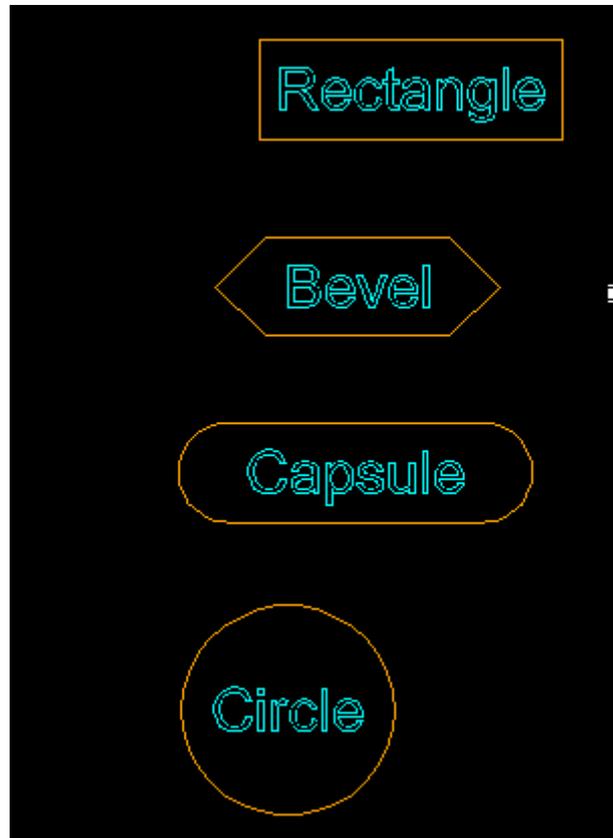
The icons at the end of the rows in Property Sheets now appear all the time. In V10 they only appeared when you clicked in the row.

For example, in **Components**



## 12.17 More Borders Types for Text

In addition to Rectangle, there are now the extra shapes Bevel, Capsule and Circle for the border around text.



## 12.18 Arithmetic in an Integer Panel Field

The arithmetic operations plus (+), subtraction (-), divide (/) and multiplication (\*) are now allowed in an Integer panel field.

The calculations are all done as real arithmetic and the result is rounded at the **end** of all the calculations and not along the way.

**Note:** An Integer panel is a panel field that only can take an integer number.

## 12.19 If Else in Real Value Fields

There is an **if else** conditional expression that can be used in **Real** fields:

`if (logical_expression) value_1 else value_2`

which has the meaning

if *the logical\_expression* is true then use *value\_1* otherwise use *value\_2*.

where **value\_1** and **value\_2** are expressions that evaluate to a real value.

One restriction is that the **logical\_expression** must be enclosed in round brackets.

For example,

`if (3*4<13) 3 else 4`

This may not seem to be of much use except that it works nicely with **Snippets** and Snippet parameters. For example, you can then have in a Real field

`if ($VAL1 < 0) -1*VAL1 else $VAL1`

where VAL1 is a snippet parameter.

### Important Note

Note that *value\_1* or *value\_2* can be an **if else** conditional expressions so you can have **nested conditional expressions**. That is

`if (logical_expression) conditional_expression_1 else conditional_expression_2`

For example

`if ($VAL1 < 0) if (3*4<13) 3 else 4 else $VAL1`

This will be parsed as

`if ($VAL1 < 0) (if (3*4<13) 3 else 4) else $VAL1`

**Hint:** if something doesn't seem to work, put more round brackets in.

## 12.20 Legacy Name in Colours.4d

There is a new column in the **colours.4d** file called **Legacy Name** and whenever a colour name is searched for, the **Legacy Name** column will also be searched after the **Colour Name** column.

The **Legacy Name** was introduced so that some of the original 12d Model colours can be given new names without upsetting the colour number that went with the original name.

Edit Colours

Found path: "C:\Program Files (x86)\12d\12dmodel\11.00\set\_ups\colours.4d"

	Colour No.	Pop-up No.	Colour Name	Legacy Name	Colour Group	Red	Green	Blue		Pen No.	Comme
24	403	-7007	pen 025		Black Pen mm weight	255	255	255		403	Plots bla
25	404	-7006	pen 025a		Black Pen mm weight	0	255	255		404	Plots bla
26	405	-7005	pen 035		Black Pen mm weight	255	255	0		405	Plots bla
27	406	-7004	pen 035a		Black Pen mm weight	0	255	0		406	Plots bla
28	407	-7003	pen 050		Black Pen mm weight	255	127	0		407	Plots bla
29	408	-7002	pen 070		Black Pen mm weight	0	0	255		408	Plots bla
30	409	-7001	pen 100		Black Pen mm weight	150	90	0		409	Plots bla
31	501	-6060	vis grass	grass	Visualisation	52	80	8		501	
32	502	-6059	vis grass1	grass1	Visualisation	74	105	11		502	
33	503	-6058	vis grass2	grass2	Visualisation	96	130	14		503	
34	538	-6057	vis grass3	grass3	Visualisation	118	155	17		538	
35	539	-6056	vis grass4	grass4	Visualisation	140	180	20		539	
36	540	-6055	vis grass5	grass5	Visualisation	162	205	23		540	
37	541	-6054	vis grass6	grass6	Visualisation	184	220	26		541	

Colour Name that is used for this colour number

Legacy Name that can also be used for this colour number

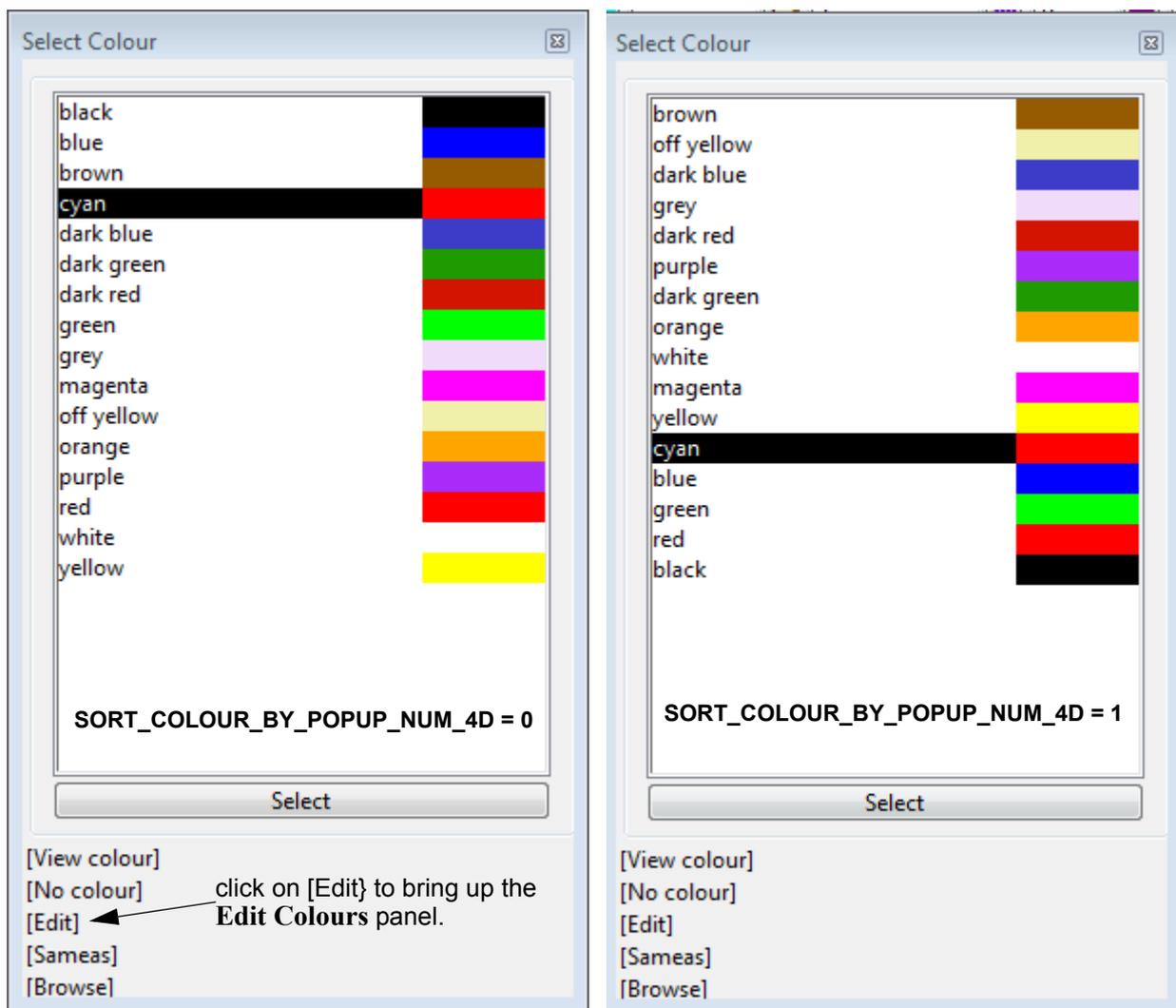
## 12.21 Colours in Pop Up Order

### 12.21.1 SORT\_COLOURS\_BY\_POPUP\_NUM\_4D

There is a new environment variable **SORT\_COLOURS\_BY\_POPUP\_NUM\_4D** that controls the display order of the colours in the **Select Colour** popup (that you get when you click on a **Colour** icon):

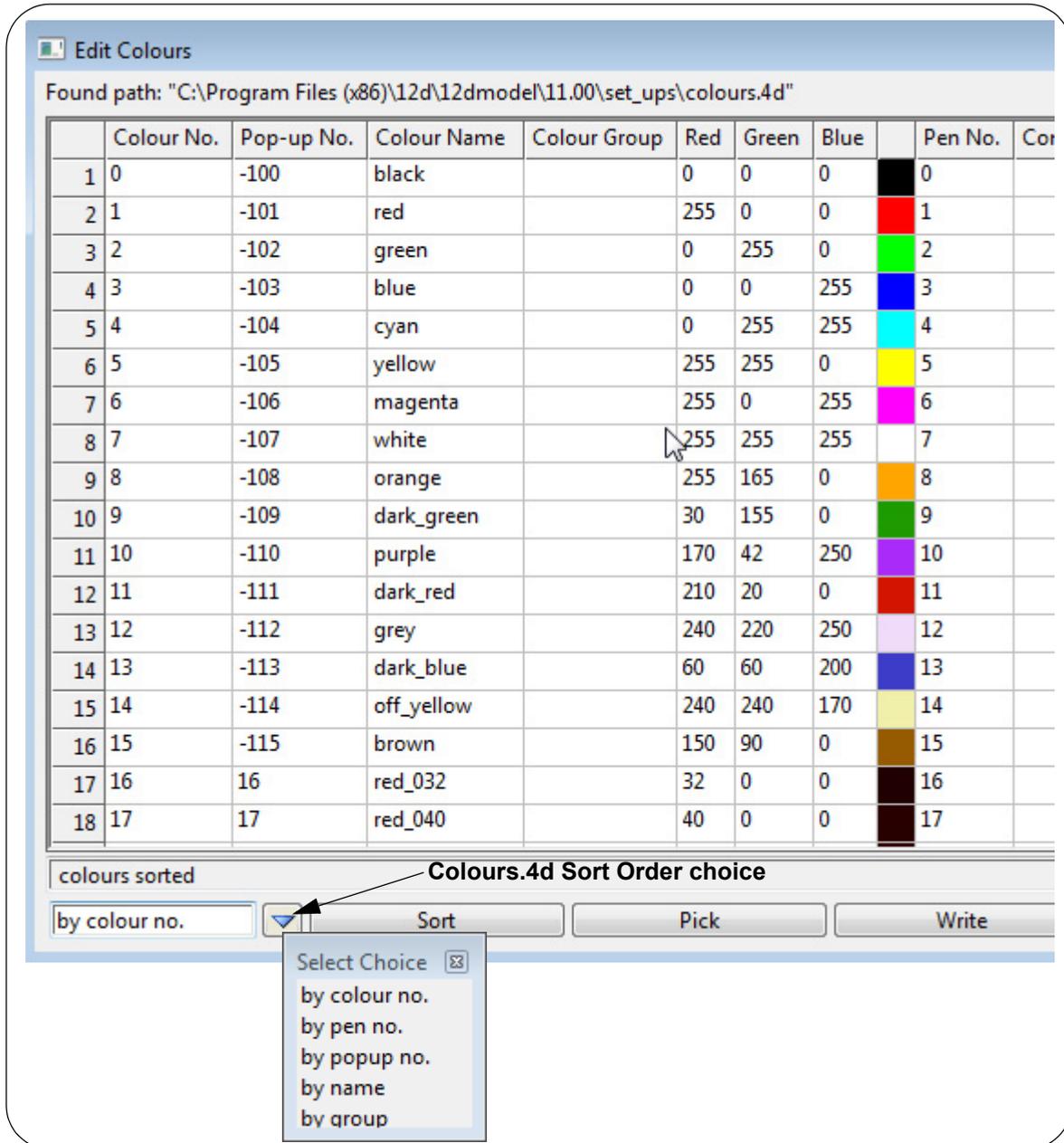
If **SORT\_COLOURS\_BY\_POPUP\_NUM\_4D** is 0, the colours in the **Select Colour** popup are sorted alphabetically.

If **SORT\_COLOURS\_BY\_POPUP\_NUM\_4D** is 1, the colours in the **Select Colour** popup are sorted by the Pop-up Number given for the colours in the **Colours.4d** file.



**Note** - Remember that the number of colours shown in the **Select Colour** popup is controlled by the **Display colours** value in the **Defaults** panel.

To edit the **Colours.4d** file to edit the popup number, click on [Edit] in the **Select Colour** popup. You can then edit the Pop-Up Number column and after any changes are made and write out the new **Colours.4d** file.



Note

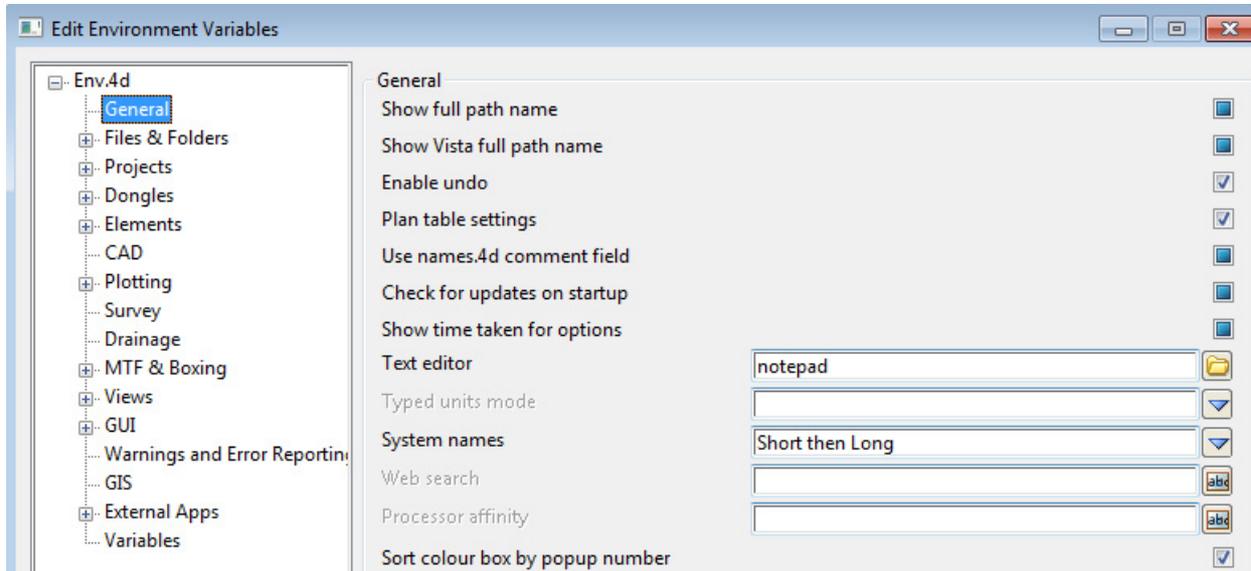
The sort order used when displaying the **Colours.4d** file has no effect on the order in the **Select Colour** pop-up.

To see the **Colours.4d** file sorted in different ways, click on the **Sort Order** choice pop-up, select the required sort order and then click on the **Sort** button.

You need to do a **project restart** for any changes in **Colours.4d** to take effect.

## 12.21.2 Setting **SORT\_COLOURS\_BY\_POPUP\_NUM\_4D**

The environment variable **SORT\_COLOURS\_BY\_POPUP\_NUM\_4D** is set in the **Edit Environment Variables** panel by the field *Sort colour box by popup number* in the **General** node.



## 12.22 Full Stop Allowed in Colour Name

Colour names can now contain a full stop (.).

For example,

green 0.25

## 12.23 Subgroups in Linestyle.4d and Symbols.4d

For the definition of a linestyle or a symbol in a linestyle.4d or symbols.4d file, the `group_name` in the keyword pair

```
group group_name
```

can define a subgroup structure for listing the name of the linestyle or symbol in the **Select Choice** popup.

The **group\_name** starts with the group and then each subgroup with the groups and subgroups being separated by a `/`.

For example,

```
group "Test/Dimension"
```

in the definition for ARROW

```
worldstyle "ARROW" {  
  xorigin 0  
  yorigin 0  
  mode vertex  
  group "Test/Dimension"
```

will show on the **Select Choice** popup for Linestyle or Symbol popup icons as



### Important Note

Even though there are groups, the names of each linestyle/symbol must be unique.

# 13. Menus on Views

See

[13.1 Text for Z Values, Point Ids etc on Plan Views](#)

[13.2 Section View 2 Points Profile](#)

[13.3 \(x,y\) in Section View](#)

[13.4 View Properties Panel](#)

[13.5 Clone a View](#)

## 13.1 Text for Z Values, Point Ids etc on Plan Views

See

[13.1.1 Text for Z Values etc on Plan Views Demystified](#)

[13.1.2 Text for Z Values etc on Plan Views - Single](#)

[13.1.3 Text for Point Id etc on Plan Views - Table](#)

### 13.1.1 Text for Z Values etc on Plan Views Demystified

**Plan View Menu =>Settings =>Z Values**

**Plan View Menu =>Settings =>Point/Vertex ids**

**Plan View Menu =>Settings =>Names**

**Plan View Menu =>Settings =>Vertex indices**

**Plan View Menu =>Settings =>Attributes**

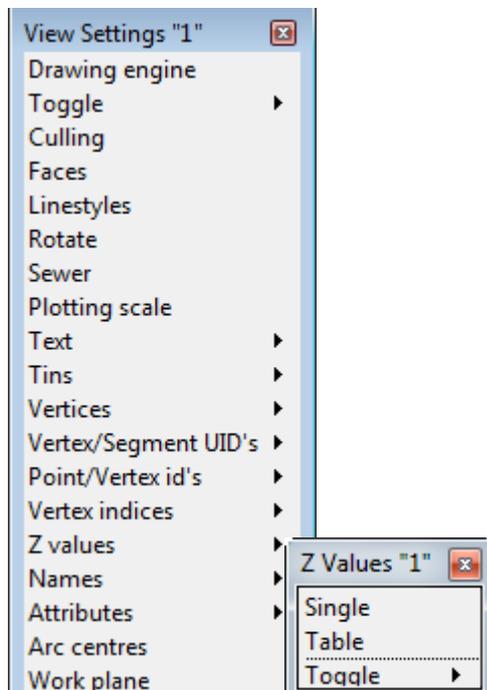
**Plan View Menu =>Settings =>Vertex/Segment UID's**

Since the addition of **Tables** to these options, a full understanding of their workings and the notorious [n/a] has baffled almost everyone. And things often stopped working the way you thought they should, and deleting the plan view was the only way out.

The root cause of the confusion was that the rows of the **Tables** only had a tick box in them, when actually they had three states. And two of the states showed up as an **off** tick box.

For V11 we have made modifications so that it should be easier to understand. So clear your minds of everything you think you know about these options and we'll try again.

We will go through the example of displaying z-values to show how it now all looks and works.



See

[13.1.1.1 Single](#)

[13.1.1.2 Table](#)

[13.1.1.3 Toggle Walk-Right](#)

### 13.1.1.1 Single

For controlling the view default for displaying of the text of z-values, **Single** can be set to draw the z-values (**on**) or don't draw the z-values (**off**).

If **Single** is set to **on**, then the z-values for all the models on the view, *except those mentioned in Table*, will be drawn.

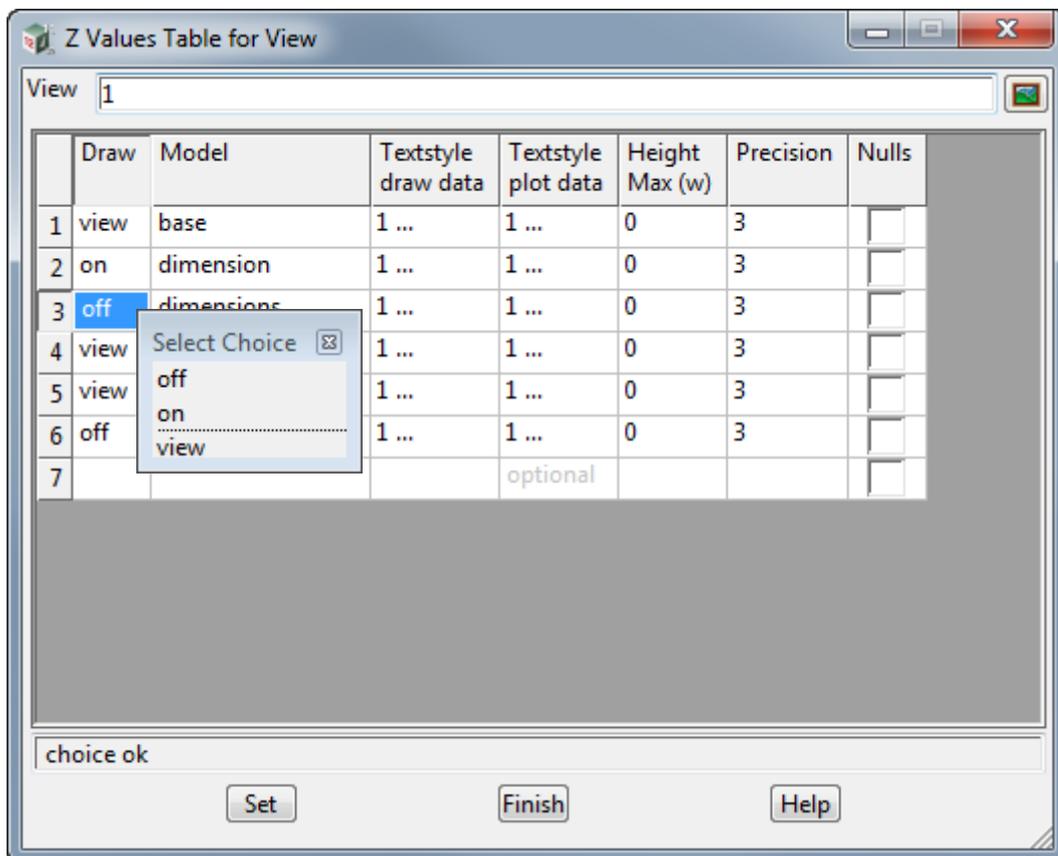
If **Single** is set to **off**, then the z-values for all the models on the view, *except those mentioned in Table*, will **not** be drawn.

### 13.1.1.2 Table

**Table** allows you to override the view default for displaying the text of z-values for specified models.

**Table** brings up a grid with zero or more rows and each row defines what is happening for a particular model. Any model not mentioned by a row takes the default **Single** setting.

Each row (and hence Model) in the **Table** has a **Draw** field with three choices - **on**, **off** and **view**.



If **Draw** is set to **on**, then if that model is on the view, the z-values for the model are **drawn regardless** of the **Single** value.

If **Draw** is set to **off**, then if that model is on the view, the z-values for the model are **not drawn regardless** of the **Single** value.

If **Draw** is set to **view**, then this row is ignored and if that model is on the view, the z-values for the model **obey** the **Single** value.

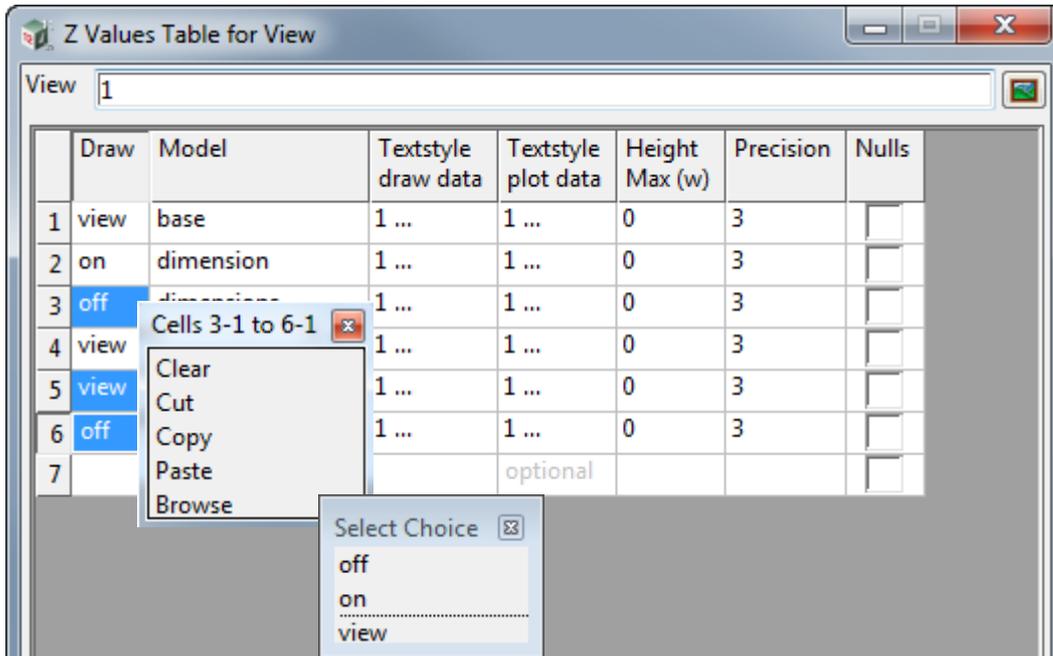
So **Table** is an override for **Single**.

To change the values in a cell in the **Draw** column, you can type into a cell, or it is better (and

safer if you can't type very well) to click **RB** over the cell and bring up the **Select Choice** menu and select **off**, **on** or **view** from the menu.

If for some reason you do not get the menu with the choices on it coming up, the menu you do get will have **Browse** on it and clicking on **Browse** will bring up the choices **off**, **on** and **view**.

To set all, or a number of the values in the **Draw** column at once, highlight the ones to change and click **RB** to bring up the **Cells** menu and select **Browse**. The choice of **off**, **on** and **view** will be made to all the highlighted cells.

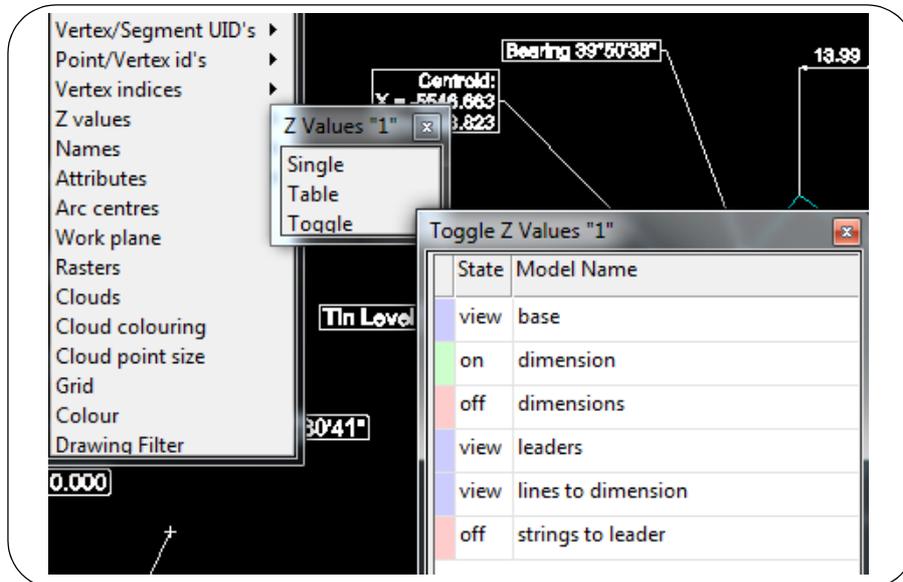


Note that clicking on the word **Draw** at the top of the **Draw** column highlights **all** the cells in the column. Holding down **<Ctrl>** and clicking **LB** when over a cell highlights/unhighlights individual cells.

If for some reason you do not get the menu with the choices on it coming up, the menu you do get will have **Browse** on it and clicking on **Browse** will bring up the choices **Draw**, **Don't draw** and **Ignore**.

### 13.1.1.3 Toggle Walk-Right

There is a **Toggles** walk right menu and it displays a list of all the recorded models in **Table** with a **State** column with either **view**, or **[off]** to indicate the draw state for that particular model.



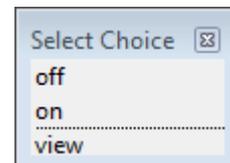
The **on**, **off** and **view** represent the value for **Draw** in the associated **Table** for the **Setting**.

If **on** then for that model in the **Table**, the **Draw** is set to **on**.

If **off** then for that model in the **Table**, **Draw** is set to **off**.

If **view** then for that model in the **Table**, **Draw** is set to **view**.

Clicking **RB** in a row in the **State** column will bring up a **Select Choice** menu.



Selecting **on**, **off** or **view** changes the value for the appropriate row in the associated **Table** and also redraws the view using the modified values.

### 13.1.1.4 Toggle Menu

Then there is the **Toggle** menu brought up by clicking on the toggle button  on a Plan view.

On the Toggle menu, there is still an **[on]** and **[off]** after the menu item to indicate the default state for the view (the value of **Single**), but there is now a **(\*)** after the **[on]** or **[off]** to indicate if there are any models in Table that are different to the **Single** state. (an **exception**)

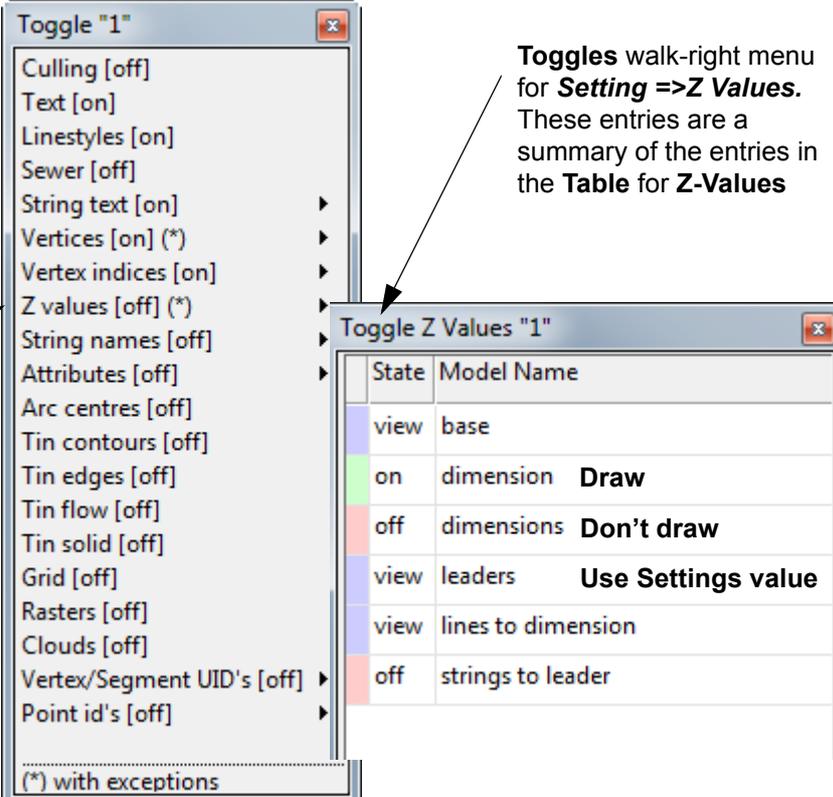
That is:

**[on]** means that the default for all models on the view is to draw the value, and that the default is applying to all the models currently on the view.

**[on] (\*)** means that the default for all models on the view is to draw the value, BUT there are some models in Table that **will not** draw the value.

**[off]** means that the default for all models on the view is to not draw the value, and that the default is applying to all the models currently on the view.

**[off] (\*)** means that the default for all models on the view is to not draw the value, BUT there are some models in Table that **will** draw the value.



The **[off]** indicates that the default for all models not mentioned in the **Table** to not have z-values labelled. This is the **Single** setting.

The **(\*)** indicates that there are some models in the **Table** that have their z-values labelled.

**Toggles walk-right menu for *Setting =>Z Values*.** These entries are a summary of the entries in the **Table** for **Z-Values**

State	Model Name
view	base
on	dimension <b>Draw</b>
off	dimensions <b>Don't draw</b>
view	leaders <b>Use Settings value</b>
view	lines to dimension
off	strings to leader

#### Warning for the Toggle Menu

Although clicking on an item in the **Toggles** menu immediately reverses the **[on]/[off]** state of the view and updates the view accordingly, the **[on]/[off]** on the **Toggles** menu does **NOT** dynamically change.

So if the **Toggles** menu is pinned up, it does not refresh itself and show the change in the **[on]/[off]** state. You only see the change if you close the **Toggle** menu and then bring it up again.

## 13.1.2 Text for Z Values etc on Plan Views - Single

**Plan View Menu =>Settings =>Z Values =>Single**

**Plan View Menu =>Settings =>Point/Vertex ids =>Single**

**Plan View Menu =>Settings =>Names =>Single**

**Plan View Menu =>Settings =>Vertex indices =>Single**

**Plan View Menu =>Settings =>Attributes =>Single**

**Plan View Menu =>Settings =>Vertex/Segment UID's =>Single**

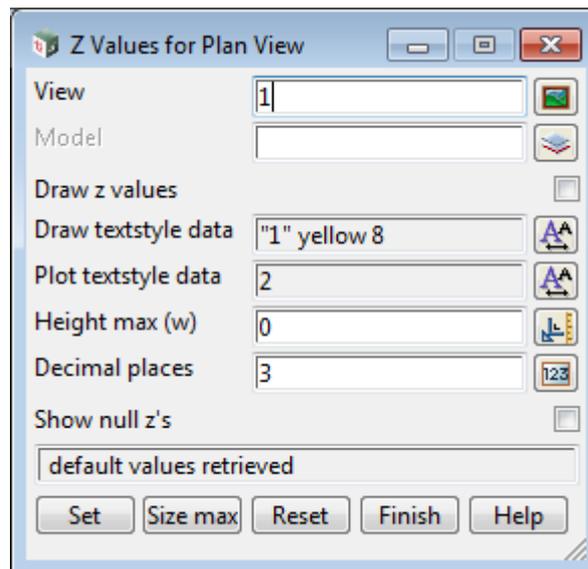
For each of the above options, all the fields for defining the drawing of the text for z-values etc on the view, for plots and for outputs, have been wrapped into a **Draw textstyle data** and a **Plot textstyle data** fields.

This means all the extra settings in a Textstyle data can now be set for drawing the text for z-values etc on a plan view.

For example, the option

**Plan View Menu =>Settings =>Z Values =>Single**

for drawing z values on a plan view, now has the panel:

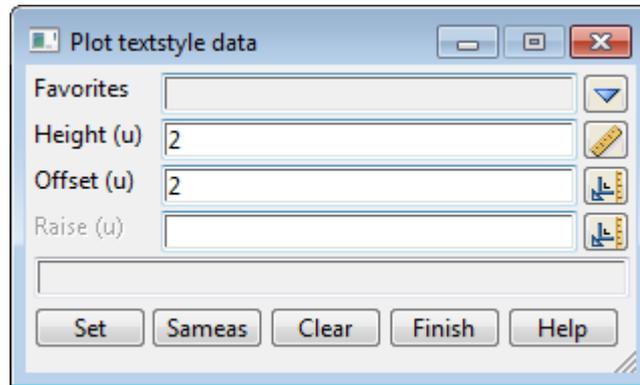


The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>View</b> <i>name of the view to set the parameters for displaying vertex z values.</i>	view box	current view	available views
<b>Model</b> <i>if not blank, then vertex z values will only be displayed for this model. If blank, then all models not mentioned in the Table option use these parameters for drawing vertex z-values on a view.</i>	model box		
<b>Draw z values</b> <i>If ticked, then vertex z values will be displayed (as long as it passes the check against Height max (w)). If not ticked, then no vertex z values for strings are displayed on the plan view.</i>	tick box		
<b>Draw textstyle data</b> <i>textstyle data to define the drawing of the z values on the view. The height, offset and raise are given in</i>	textstyle data box		available textstyle datas

screen units (pixels).

**Plot textstyle data**                      plot textstyle data box                      available textstyle datas  
 a plot textstyle data to define the height, offset and raise values to use when the data is plotted or output (using the File =>Data output options).  
 When the data is plotted, the units are millimetres (paper size).  
 When the data is output, the units are world (world size).



**Height max (w)**                      real box                      0  
 when you have a plan view then for the current view settings, any text drawn at a screen size (pixels) would have an equivalent size in world units.  
 As you zoom in, the equivalent size in world units will decrease. That is, the screen texts gets smaller in equivalent world units.  
 As you zoom out, the equivalent size in world units increases. That is, the screen texts gets larger in equivalent world units.  
 If Height max (w) is non-zero then when the zoom settings on the plan view is such that the equivalent height of the z values in world units is **greater** than Height max (w), then the z values are not drawn. Consequently as one zooms out, the z values will eventually stop drawing.  
 If Height max (w) is zero, the z values are always drawn.

**Decimal places**                      real box                      3  
 the number of decimal places to use when displaying the z values.

**Show null z's**                      tick box  
 if **ticked**, null z values are displayed as the text "null".  
 If **not ticked**, no z value is displayed at null z-values.

**Set**                                      button  
 set the values in the panel and then redraw the plan view.

**Size max**                              button  
 when clicked, the **Height max (w)** field is given the current equivalent world size of the vertex z-values displayed on the given plan view. Hence if any further zoom out is done, the vertex z-values will stop drawing.  
 After clicking **Size max**, the **Set** button must then be clicked to set the new **Height max (w)** for the view.

**Reset**                                      button  
 reset all the parameters to the default for plan views.

### 13.1.3 Text for Point Id etc on Plan Views - Table

- Plan View Menu =>Settings =>Text =>Table**
- Plan View Menu =>Settings =>Vertices =>Table**
- Plan View Menu =>Settings =>Vertex/Segment UID's =>Table**
- Plan View Menu =>Settings =>Point/Vertex ids =>Table**
- Plan View Menu =>Settings =>Vertex indices =>Table**
- Plan View Menu =>Settings =>Z Values =>Table**
- Plan View Menu =>Settings =>Names =>Table**
- Plan View Menu =>Settings =>Attributes =>Table**

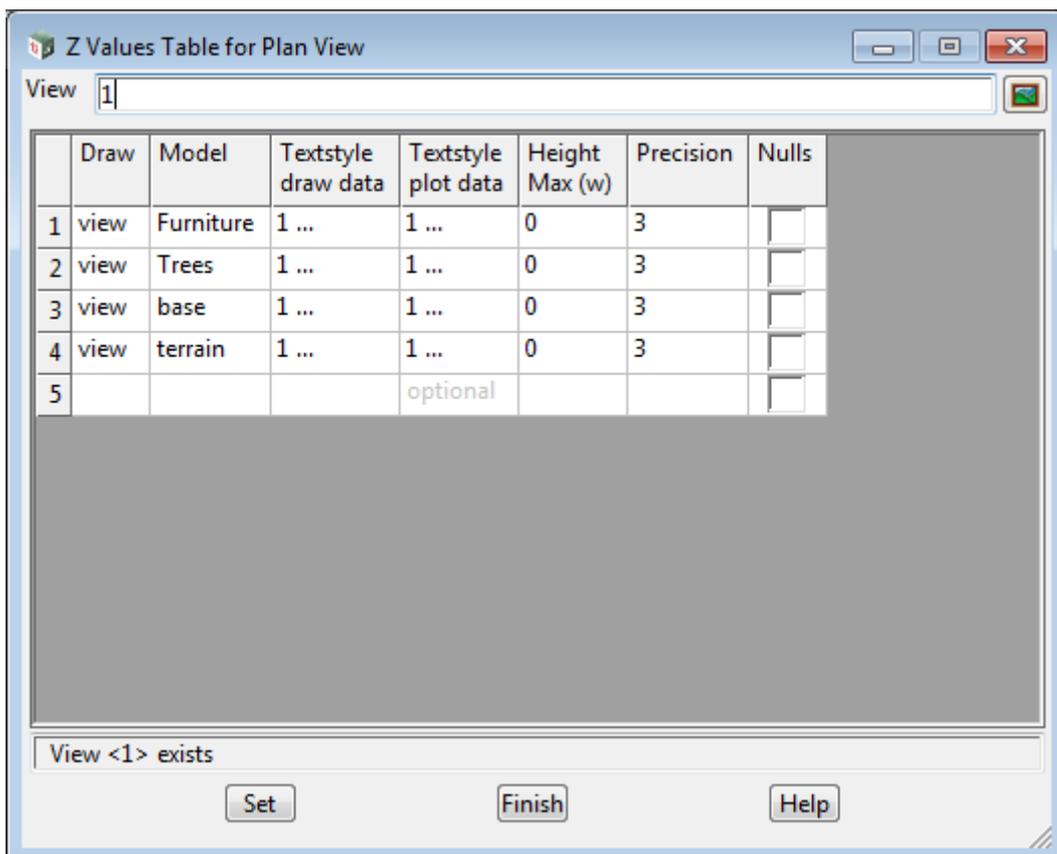
For each of the above options, for each given model, all the fields for defining the drawing of the text for z-values etc on the view, for plots and for outputs, have been wrapped into a **Draw textstyle data** and a **Plot textstyle data** fields in the row for that model in the grid.

This means for each model, all the extra settings in a Textstyle data can now be set for drawing the text for z-values etc on a plan view.

For example, the option

- Plan View Menu =>Settings =>Z Values =>Table**

for drawing z values on a plan view, now has the panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>View</b>	view box	current view	available views

*name of the view to set the parameters for displaying vertex z values.*

## Grid

**Draw** choice box Don't draw, Draw, Ignore

If **Draw** is set to **on**, then if that model is on the view, the Values Text for the model are **drawn** regardless of the **Single** value.

If **Draw** is set to **off** then if that model is on the view, the Values Text for the model are **not drawn** regardless of the **Single** value.

If **Draw** is set to **view**, then if that model is on the view, the Values Text-for the model **obey the Single** value. This is the same as the row not being in the table.

So the **Draw** column in **Table** is an override for **Single**.

**Model** model box

if **not blank**, then the display of vertex z values for the strings in this model will obey the choice for this row of the **Draw** column.

**Textstyle draw data** textstyle data box available textstyle datas

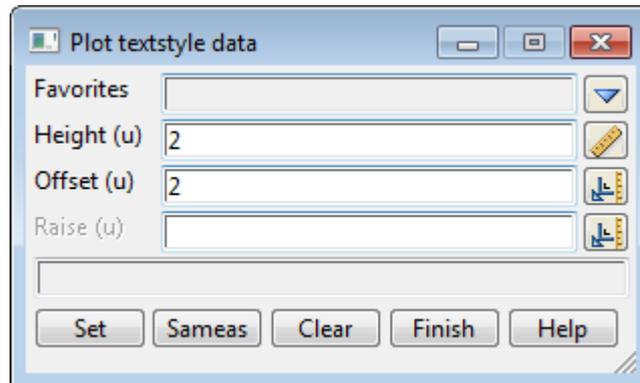
textstyle data to define the drawing on the view of the z values in the model. The height, offset and raise are given in screen units (pixels).

**Textstyle plot data** plot textstyle data box available textstyle datas

a plot textstyle data to define the height, offset and raise values to use when the data in the model is plotted or output (using the File =>Data output options).

When the data in the model is plotted, the units are millimetres (paper size).

When the data in the model is output, the units are world (world size).



**Height max (w)** real box 0

when you have a plan view then for the current view settings, any text drawn at a screen size (pixels) would have an equivalent size in world units.

As you zoom in, the equivalent size in world units will decrease. That is, the screen texts get smaller in equivalent world units.

As you zoom out, the equivalent size in world units increases. That is, the screen texts get larger in equivalent world units.

If Height max (w) is non-zero then when the zoom settings on the plan view is such that the equivalent height of the z values in world units is **greater** than Height max (w), then the z values are not drawn.

Consequently as one zooms out, the z values will eventually stop drawing.

If Height max (w) is zero, the z values in the model are always drawn.

**Precision** integer box 3

the number of decimal places to use when displaying the z values in the model.

**Nulls** tick box

*if ticked, null z values in the model are displayed as the text "null".*

*If not ticked, no null z-value in the model are displayed.*

**Set** button

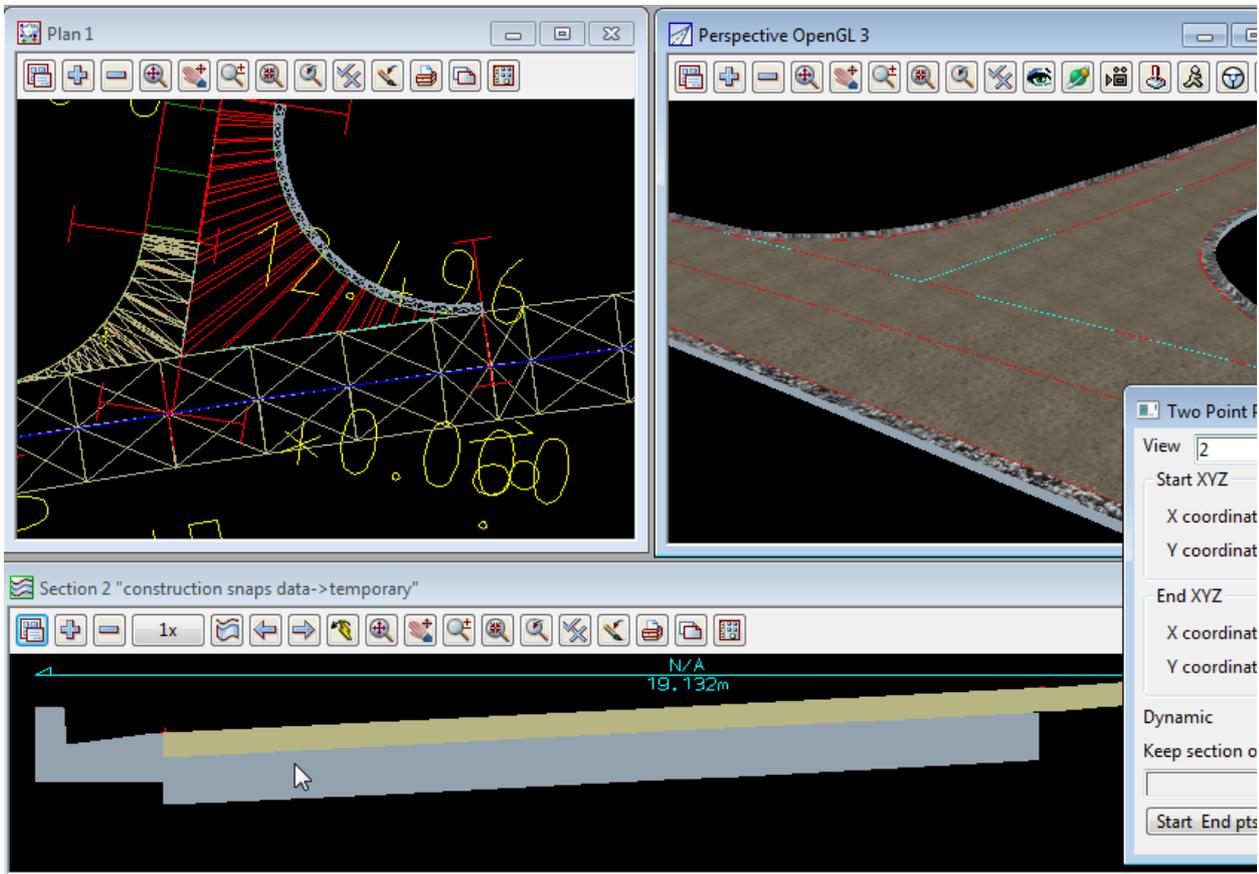
*set the values in the panel and then redraw the plan view.*

## 13.2 Section View 2 Points Profile

Plan View Menu =>Settings =>Profile =>2 Points

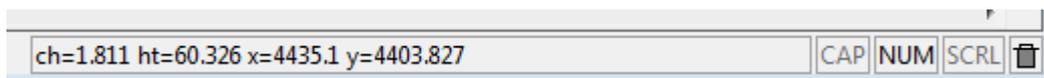
This option will now section through extrusions, trimeshes and meshes.

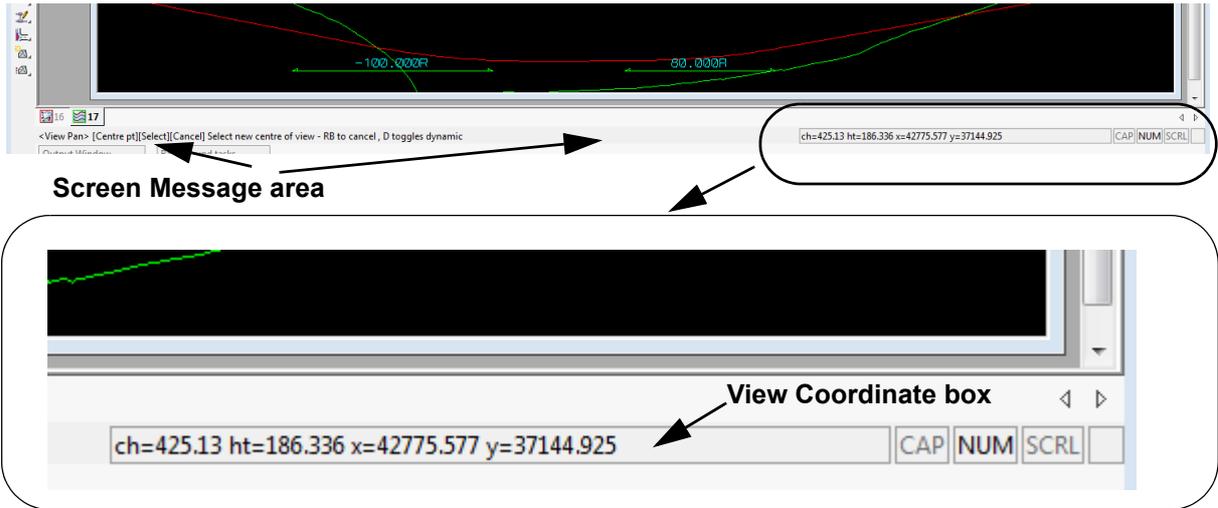
For example sectioning through trimeshes produced by an Apply MTF gives:



## 13.3 (x,y) in Section View

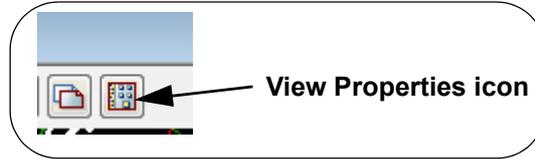
When in a Section view that has a string profiled on it, the (x,y) coordinates of the cursor position are also displayed with the (chainage, height) in the **View Coordinate** box in the **Screen Message Area** at the bottom of the **12d Model** window.





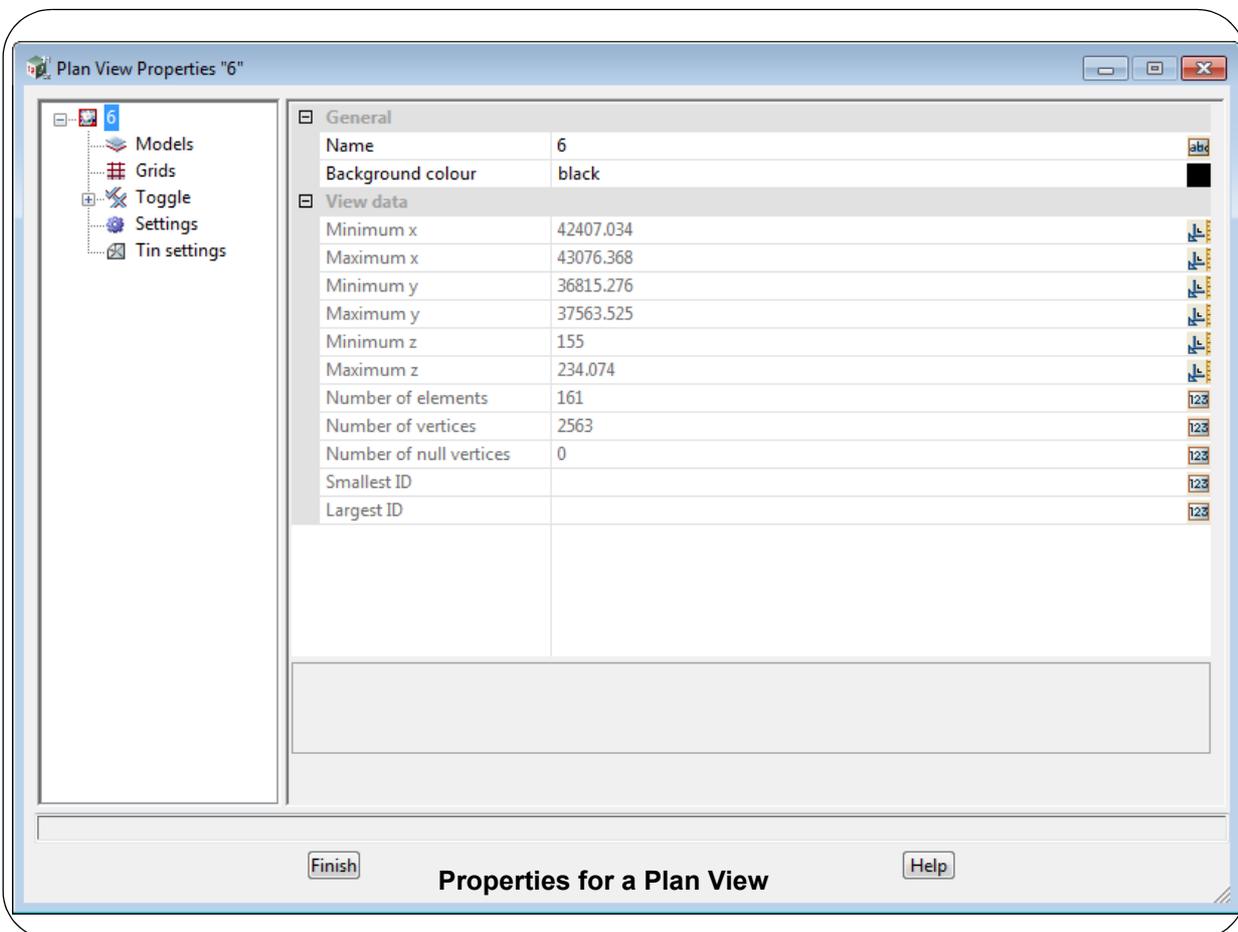
# 13.4 View Properties Panel

On each View, there is now an icon in the View Buttons area to bring up a **View Properties** panel.



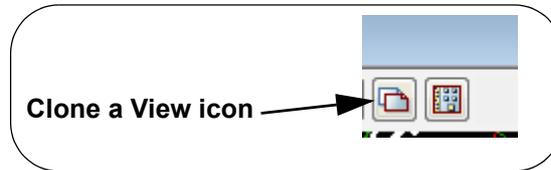
Clicking on the **View Properties** icon brings up the **View Properties** panel for that view and view type.

Values in the **View Properties** panel can be changed and the changes take place immediately.



## 13.5 Clone a View

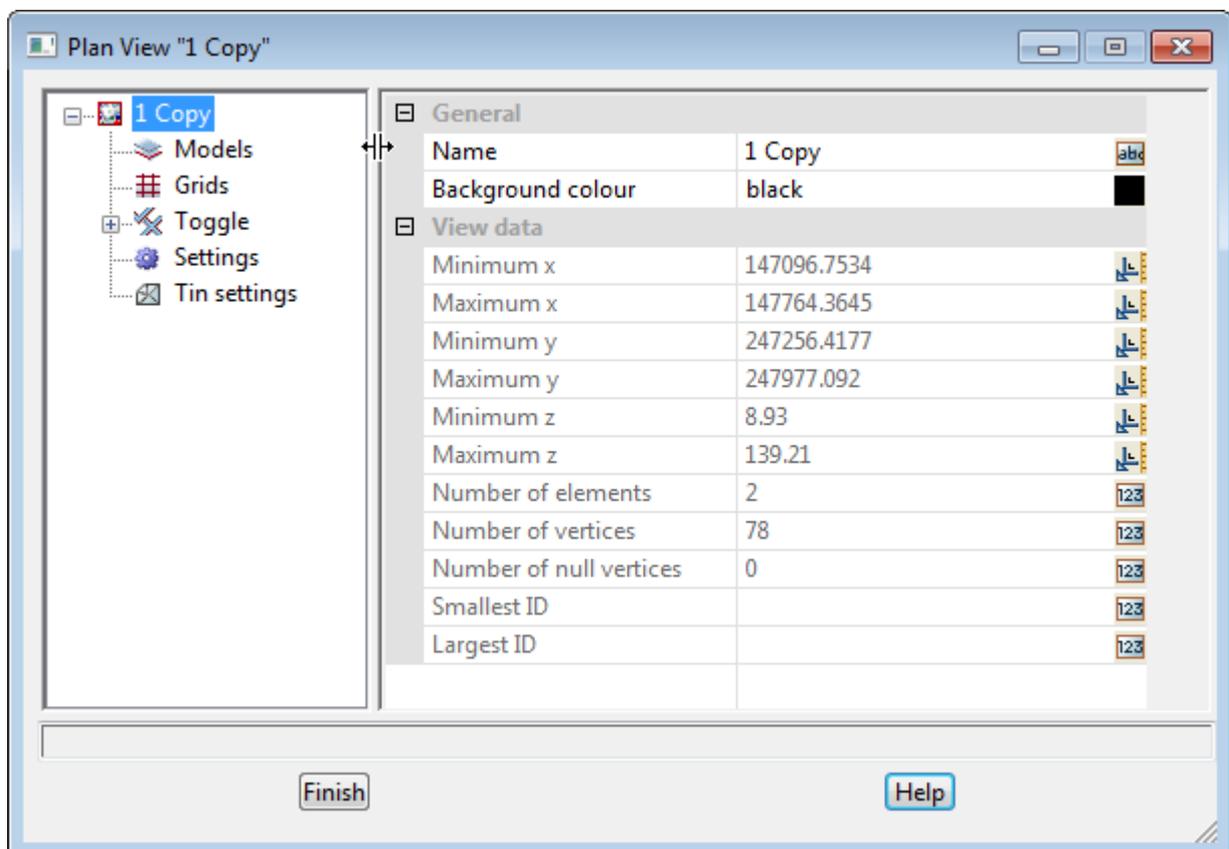
On each View, there is now an icon in the View Buttons area to **Clone a View**.



Clicking on the **Clone** icon will create a new view identical to the original view with all the same models on the new plus and apart from the view name, all the new view settings are identical to the original view.

The view name of the Cloned view is the name of the original view with " Copy" appended to it.

A **Plan View Properties** panel the new view is also brought up so that any of the View properties (for example the view name) can be quickly changed.



# 14. Project Menu

See

[14.1 Projects =>Utilities =>Zip Removed](#)

[14.2 New Projects =>Recent projects](#)

[14.3 Modified Projects =>Open](#)

[14.4 Modified Projects =>New](#)

[14.5.1 Modified Defaults](#)

[14.5.2 Modified env.4d](#)

## 14.1 *Projects => Utilities => Zip Removed*

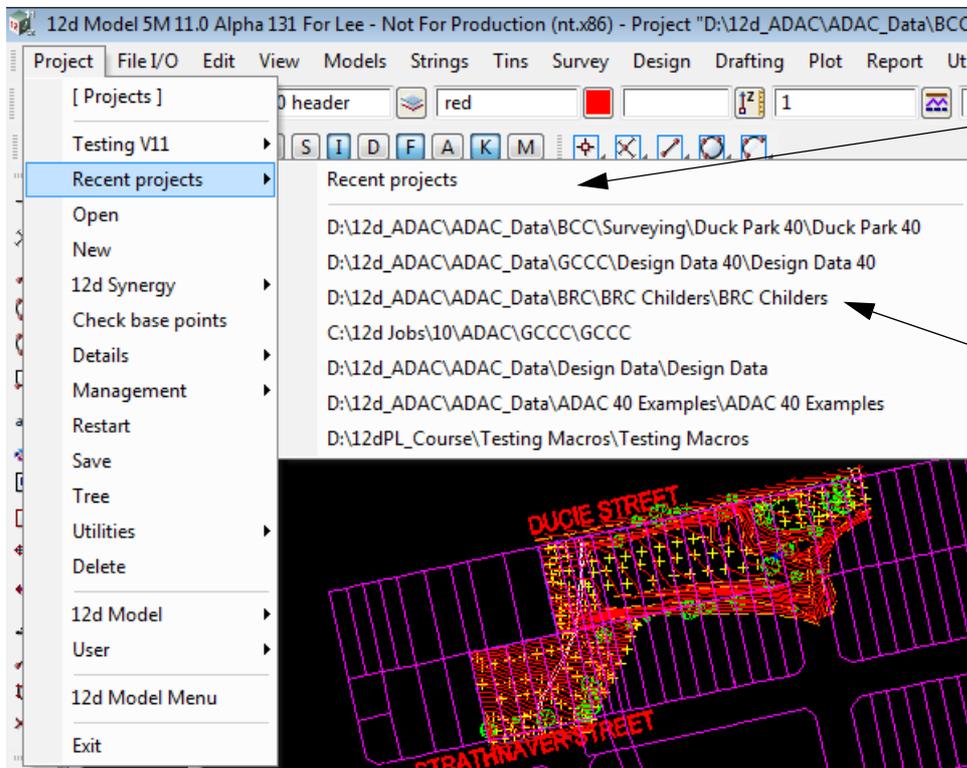
This option has been removed.

There is a zip project option and icon on the new front panel, **Open a Recent Project**.

## 14.2 *New Projects => Recent projects*

Walking right on the **Project =>Recent projects** list the recent projects and clicking on a project in the list will open the project.

Clicking on the **Recent Projects** heading on the **Main Menu** or on the **Recent projects** item when the **Projects** menu is pinned up, brings up the new **Front** panel which shows the recent project in a panel [2.1 New Front Panel](#).



Click on here to bring up the Recent Project panel

Click on a project name to open the project

## 14.3 *Modified Projects => Open*

Clicking on **Open** brings up the **Open** tab on the new **Front** panel. See [2.2 Browse Button and Open Tab](#).

## 14.4 *Modified Projects => New*

Clicking on **New** brings up the **New** tab on the new **Front** panel. See [2.3 New Button and New Tab](#).

## 14.5 Management

See

[14.5.1 Modified Defaults](#)

[14.5.2 Modified env.4d](#)

[14.5.3 Dongles](#)

[14.5.4 Managers](#)

[8.2 Share Manager](#)

[14.5.5 Toggle Topmost Buttons](#)

## 14.5.1 Modified *Defaults*

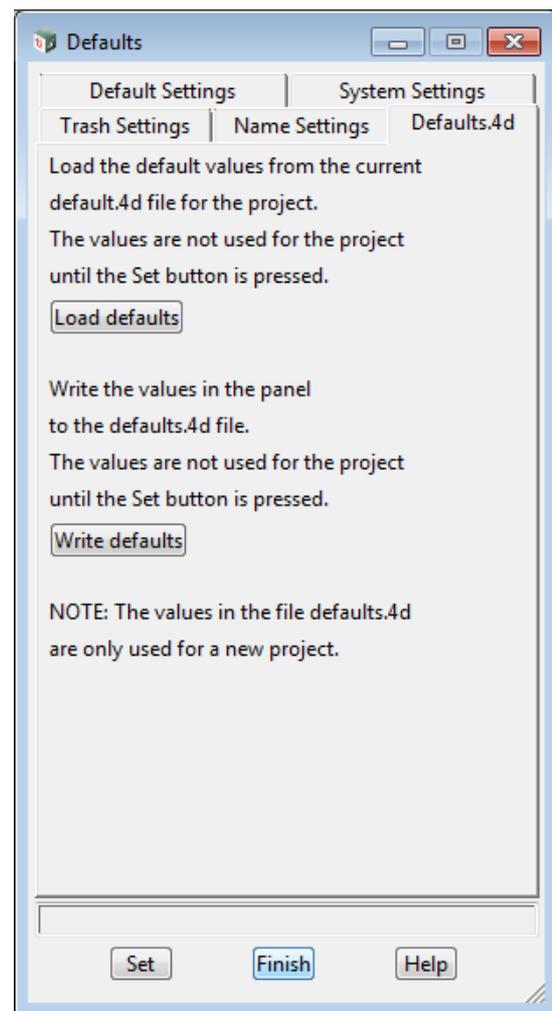
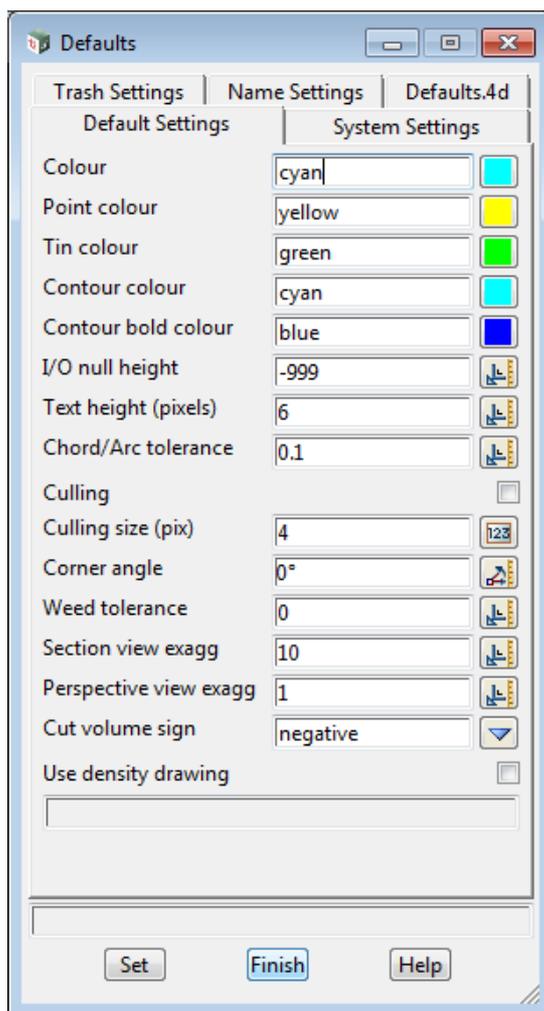
### Project =>Management =>Defaults

The **Defaults** panel has been rearranged.

The **Load** and **Write** buttons have been removed from the bottom of the panel and placed on the tab **Defaults.4d** and their names have changed to **Load defaults** and **Write defaults** respectively.

Some of the text from the *Help* has also been added to the **Defaults.4d** tab so that the user definitely knows that the *defaults.4d* file is only used when creating a new project.

Pressing the **Set** button also notifies the project that a **Save** needs to be done.



## 14.5.2 Modified *env.4d*

The env.4d panel has been rearranged and there are a number of new environment variables as well. The Edit Environment Variables panel is brought up by clicking on

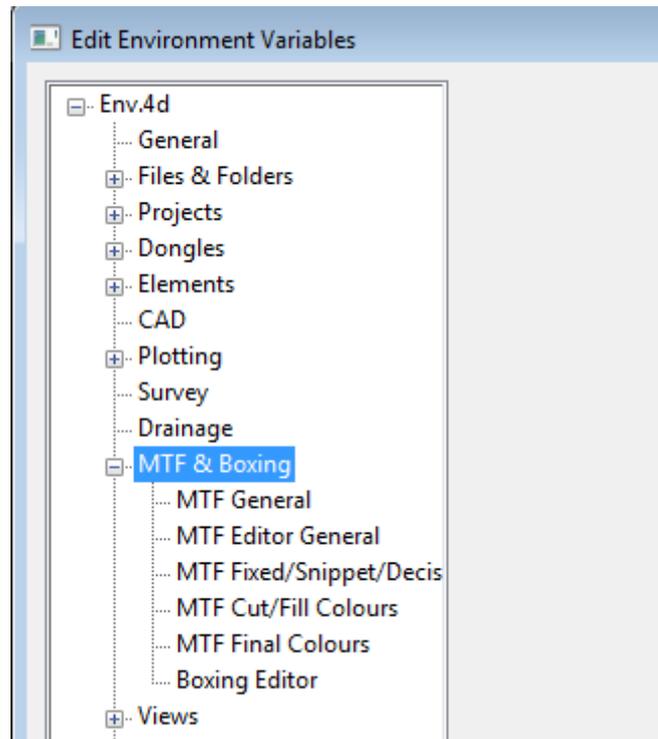
**Project =>Management =>env.4d**

See

[14.5.2.1 MTF and Boxing](#)

### 14.5.2.1 MTF and Boxing

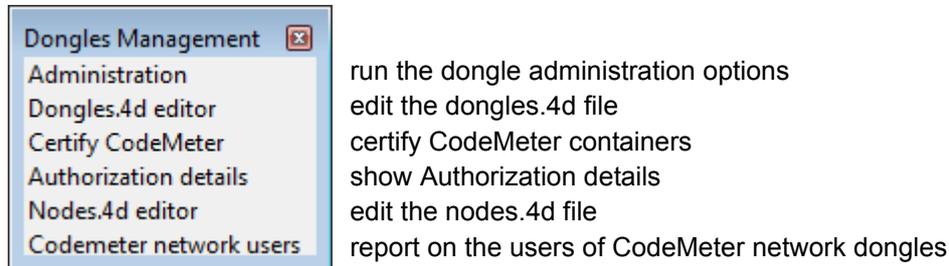
The MTF and Boxing environment variables are now in the one node **MTF & Boxing**.



## 14.5.3 Dongles

**Position of menu:** Project =>Management =>Dongles

The **Dongles** walk-right menu contains options to discover all the dongles on your subnet, edit the `dongles.4d` and `nodes.4d` files, certify CodeMeters and show authorization information.



See

[14.5.3.2 Dongles Administration](#)

[14.5.3.3 Dongles.4d Editor](#)

[14.5.3.4 Certify CodeMeter](#)

[14.5.3.5 Authorization Details](#)

[14.5.3.6 Nodes.4d Editor](#)

[14.5.3.7 CodeMeter Network User](#)

Also for information on the **Authorization Error** panel that comes up if there is an error authorising **12d Model**, see [14.5.3.5.1 Authorization Error](#).

For more information about what is needed for authorizing 12d Model, see [14.5.3.1 12d Dongles, Dongles.4d, Nodes.4d, and Authorizing](#).

### 14.5.3.1 12d Dongles, Dongles.4d, Nodes.4d, and Authorizing

To control the licensing of 12d Model and specify what modules are used for each 12d Model license, 12d Model uses a combination of a **physical hardware lock** and a **nodes.4d file**.

The **physical hardware lock** is a **Wibu dongle** or a **CodeMeter Container**.

A **Wibu dongle** is either a Wibu USB Standalone (local) dongle (green colour), or a Wibu USB Network dongle (blue colour) that controls a given number of licenses. Each Wibu dongle has a unique 12d Model dongle number. See [14.5.3.1.1 Wibu Dongles](#)

A **CodeMeter Container** comes in a variety of physical shapes (USB, SD card etc) but unlike the Wibu dongle, the CodeMeter Container can contain one or more *virtual* 12d Model dongle number. Each virtual 12d Model dongle number is either a Standalone number or a Network number, and in Network case, it also have a given number of licenses. See [14.5.3.1.2 CodeMeter Containers](#).

Hence the **12d Model dongle number** (from a physical Wibu dongle or within a CodeMeter Container) is either

a Standalone (Local) number

or

a Network number with a given number of licenses

For brevity, the **12d Model dongle number** is also referred to as the **12d dongle number** or just the **12d dongle**. But remember that the **12d dongle** may not be a physical item - it may be just one of many 12d dongles stored in a CodeMeter Container.

So the term **12d dongle** will be used to mean:

- (a) a physical Wibu dongle (either stand alone or network). This has a label with the 12d dongle number written on it.
- (b) a virtual 12d dongle inside a CodeMeter Container. The 12d dongle numbers are obtained by running a CodeMeter Administration tool.

The **dongles.4d** file lists the computers to search for Wibu dongles and CodeMeter Containers to find 12d dongle numbers. It also defines the order to search for both computers and dongle types. See [14.5.3.3 Dongles.4d Editor](#).

The **nodes.4d** file is supplied by your 12d Model Reseller and contains a list of 12d Model dongle numbers that are valid for you, and for each 12d Model dongle number in the list, the valid modules for that 12d Model dongle number.

When 12d Model tries to create a new project, or open an existing project, 12d Model uses

1. the **dongles.4d** file to obtain a usable 12d Model dongle number from the specified Wibu dongles or CodeMeter Containers  
and if a 12d Model dongle number is found,
2. a matching valid entry in the **nodes.4d** file for that 12d Model dongle number.

If a match is found then the **12d Model** project is opened.

Otherwise an **Authorization Error** panel is displayed.

See

[14.5.3.1.1 Wibu Dongles](#)

[14.5.3.1.2 CodeMeter Containers](#)

### 14.5.3.1.1 Wibu Dongles

A **Wibu dongle** can be either a Wibu USB Standalone (local) dongle (a translucent green colour), or a Wibu USB Network dongle (a solid colour) that controls a given number of licenses. Each Wibu dongle has a unique 12d Model dongle number.

The 12d Model dongle numbers for Wibu Standalone (Local) dongles start with 572d.  
The 12d Model dongle numbers for Wibu Network dongles start with e151.

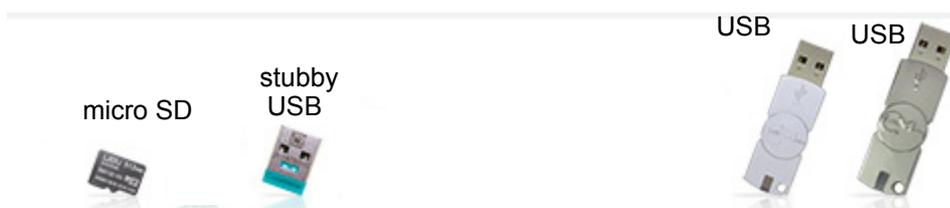


Continue to the next section [14.5.3.1.2 CodeMeter Containers](#) or return to [14.5.3.1 12d Dongles, Dongles.4d, Nodes.4d, and Authorizing](#) or [14.5.3 Dongles](#).

### 14.5.3.1.2 CodeMeter Containers

In V11 a new type of physical hardware lock, referred to as a **CodeMeter Container**, has been introduced.

The **CodeMeter Containers** come in a variety of physical shapes - USB, stubby USB and micro SD - to better suite the type of computer you are using.



Unlike the Wibu dongle, the CodeMeter Container can contain one or more *virtual* 12d Model dongle numbers.

Each virtual 12d Model number may be either a Standalone number or a Network number, and in the Network case, it will also have a given number of licenses.

The 12d Model Local (Standalone) dongle numbers in a CodeMeter Container start with.  
The 12d Model Network dongle numbers in a CodeMeter Container start with ec51.

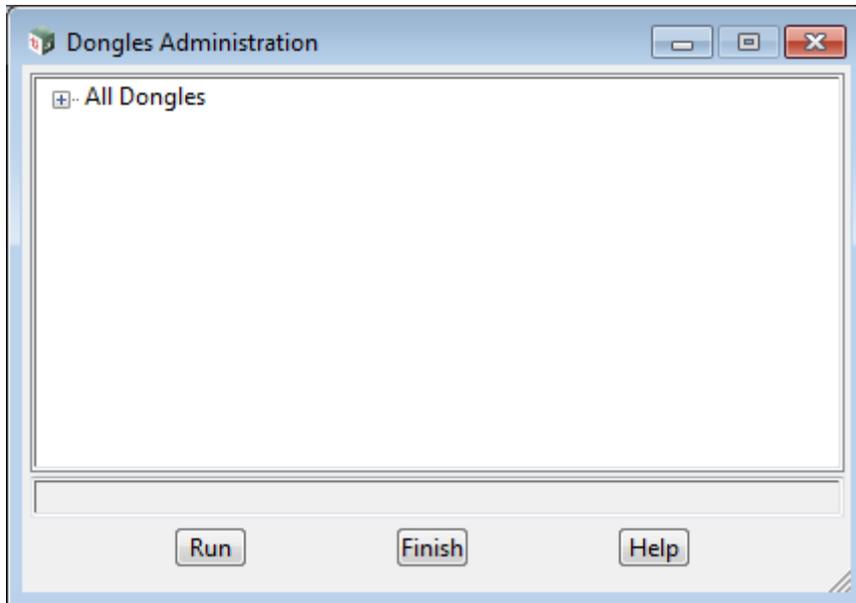
Return to [14.5.3.1 12d Dongles, Dongles.4d, Nodes.4d, and Authorizing](#) or [14.5.3 Dongles](#).

### 14.5.3.2 Dongles Administration

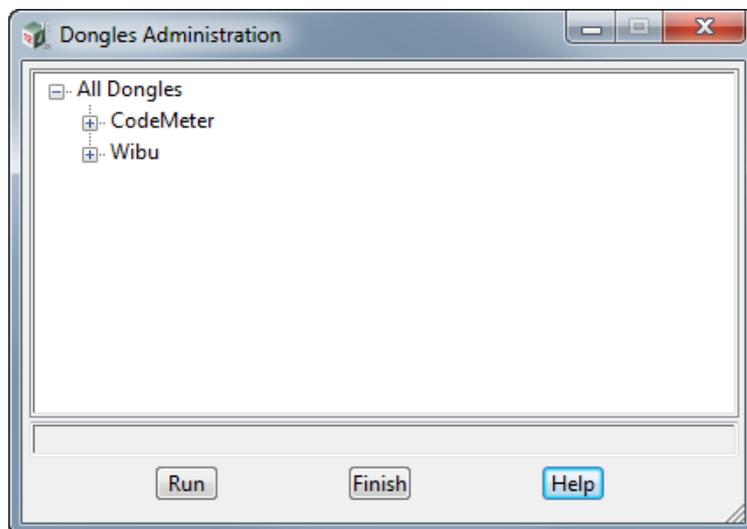
Position of option on menu: Project =>Management =>Dongles =>Administration

Position of option on menu: Help =>Dongles

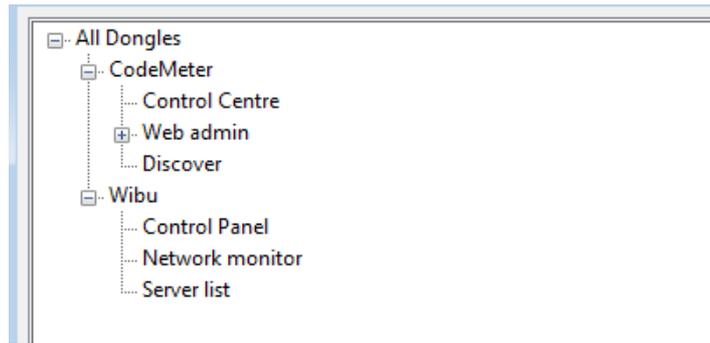
Selecting Administration brings up the **Dongles Administration** panel which displays information on **CodeMeter Containers** and **Wibu Dongles** as nodes in a tree structure.



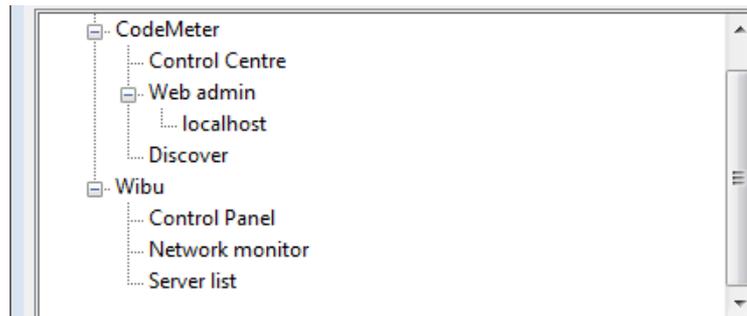
Clicking on the + in front of **All Dongles** will list the **physical dongles** that could be on your computer. That is, **CodeMeter Containers** and **Wibu Dongles**.



The **CodeMeter** and **Wibu** nodes can also be expanded by clicking on their +.



Clicking + on **CodeMeter >Web admin** will show the computer you are on as **localhost**.

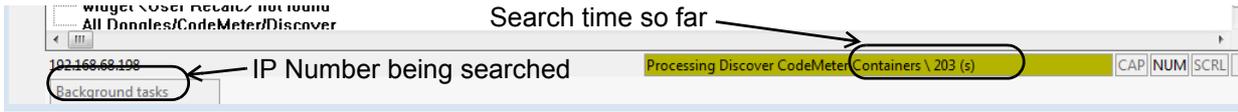


1. Clicking on **CodeMeter >Web admin >localhost** will run the CodeMeter Administration program and display information about the computer you are on in **Home** tab of the **CodeMeter WebAdmin** panel.

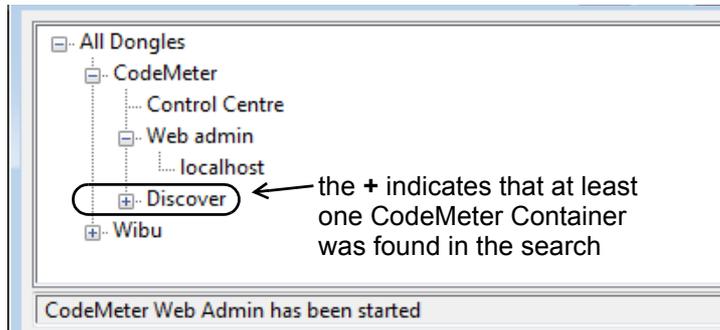


2. Clicking on **CodeMeter >Discover** will search the 255 sub nets looking for any *CodeMeter* Containers on each subnet.

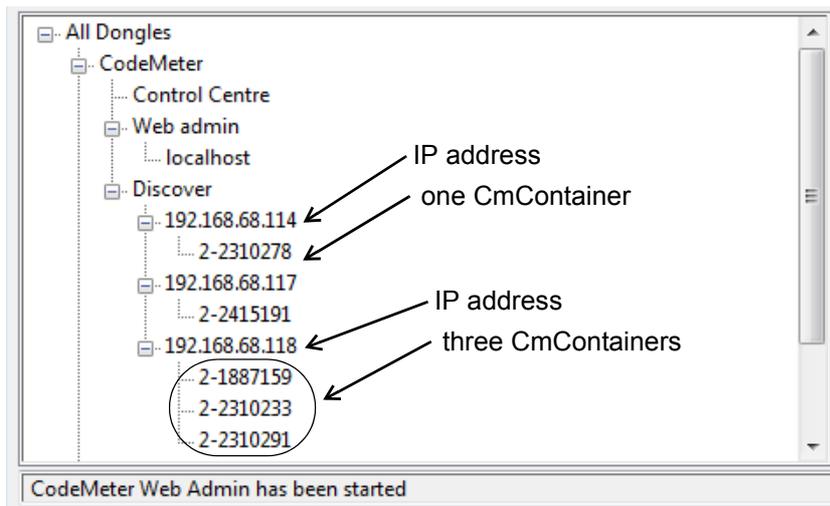
The **Discovery** may take up to five minutes and you can see the progress of the **Discovery** as it lists the IP addresses being searched in the left had corner of the screen message area, and the right hand side displays how many seconds the **Discovery** has been running.



When **Discovery** has finished searching the 255 sub nets, the IP address of any computer with a CodeMeter Container on it is listed under the **Discover** node. So a **+** will appear in front of **Discover** if at least *one* CodeMeter Container has been found.



Expanding **Discover** will display the IP Addresses with *CodeMeter* Containers (called **CmContainers**) on them. And expanding an IP Address will list all the CmContainers attached to that IP.



Double clicking on a **CmContainer** will bring up the **CodeMeter WebAdmin** panel with the **Content >Licenses** tab on display.

The **Content >Licenses** tab will show the **Products** it contains and for each **12d Product**, it will show the

- (a) 12d Product name
- (b) 12d Customer name
- (c) 12d Model dongle number (a virtual dongle inside the physical CodeMeter Container)
- (d) **number of licenses** for that **12d Model dongle**.

**CodeMeter WebAdmin**

Home | Content | **Server** | Configuration | Diagnosis | Info | Help

CmContainer | Licenses | User Data | Backup/Restore

CmContainer: 2-2310278

100003   Bundling Articles					
Product Code	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
1	SecuriKey Lite	n/a	n/a	n/a	1
101956   12d.com					
Product Code	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
1	12d Model - 12d Solutions - 5c2d47013c	n/a	n/a	n/a	1

12d Product name      12d Dongle number      12d Customer name      Number of licenses on the 12d dongle

Note that there may be more than one virtual **12d Dongle** inside the one physical CodeMeter Container, as well as products from other software vendors. For example,

**CodeMeter WebAdmin**

Home | Content | **Server** | Configuration | Diagnosis | Info | Help

CmContainer | Licenses | User Data | Backup/Restore

CmContainer: 2-2310233

100003   Bundling Articles					
Product Code	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
1	SecuriKey Lite	n/a			
101956   12d.com					
Product Code	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
1	12d Model - Stock - 12d Solutions - ec51470113	n/a	n/a	n/a	5
1	12d Model - Stock - 12d Solutions - ec51470114	n/a	n/a	n/a	25

two 12d Model products      Number of licenses on each virtual 12d dongle

- Other information can be displayed by clicking on the other tabs and subtabs of the **CodeMeter WebAdmin** panel.

Continue to the next section [14.5.3.3 Dongles.4d Editor](#) or return to [14.5.3 Dongles](#).

### 14.5.3.3 Dongles.4d Editor

**Position of option on menu:** Project =>Management =>Dongles => Dongles.4d editor

The **dongles.4d** file is new in V11 and it replaces the **Wibu** network environment variables in env.4d.

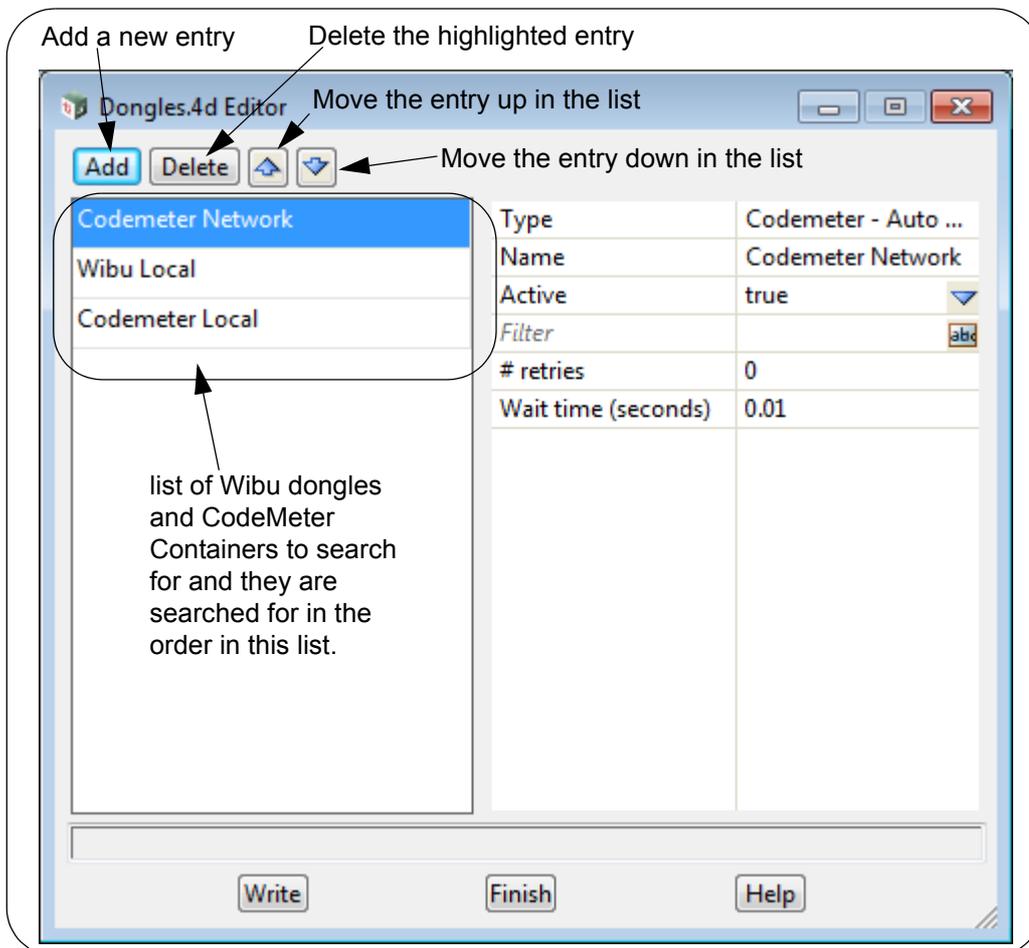
**However** the **dongles.4d** file shipped with **12d Model** contains no references to network dongles and in that case, the WIBU\_DONGLE\_4D and WIBU\_IPADDR network environment variables are still used.

So people using Wibu networks dongles **will not need to do anything** for V11 to work as before **until** they start using Codemeter Containers.

The **dongles.4d** file lists the computers to search for Wibu dongles ([14.5.3.1.1 Wibu Dongles](#)) and CodeMeter Containers ([14.5.3.1.2 CodeMeter Containers](#)) to find 12d dongle numbers. It also defines the order to search for both computers and dongle types.

The **Dongles Editor** option is for editing the **dongles.4d** file.

Selecting **Dongles.4d editor** brings up the **Dongles.4d Editor** panel.

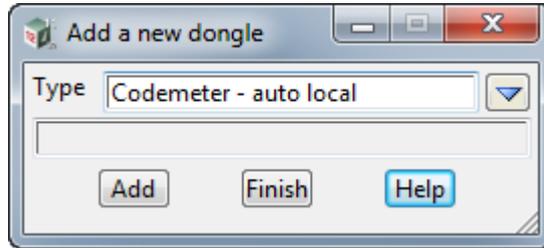


The **Dongles.4d Editor** panel displays the list of Wibu dongles (local and/or network) and CodeMeter containers (local and/or network) to search for to find 12d dongles to potentially use when opening a project. The order of searching is the order that the items occur in this list.

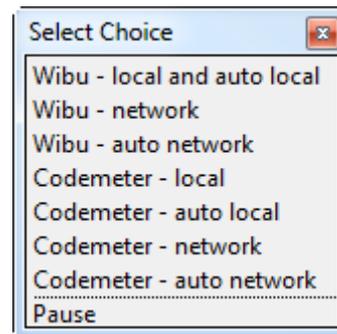
The fields and buttons used in this panel have the following functions.

Field Description                      Type                      Defaults                      Pop-Up

**Add** button  
*this brings up the Add a New Dongle panel*



**Type** choice box



**Wibu - local and auto local:** search the local computer for a Wibu Standalone dongle. If there is more than one Standalone Wibu dongle on the computer then the first one found is used. So there should only be one Standalone Wibu dongle on the computer.

	Type	Wibu - Local and Auto Local
user defined name	Name	Wibu Auto Local
use entry if Active is true	Active	true
number of times to retry	# retries	0
time to wait between retries	Wait time (seconds)	0.01

**Wibu - network:** a server name (computer name) is given as part of the information for this Type. The computer on the network with this name is searched for a Wibu Network dongle. There should only be one Wibu network dongle on the given computer

	Type	Wibu - Network
user defined name	Name	New dongle
use entry if Active is true	Active	true
name of the computer	Server name	
number of times to retry	# retries	0
time to wait between retries	Wait time (seconds)	0.01

**Wibu - auto network:** search all computers on the network for Wibu Network dongles. Note there

should only be one Wibu network dongle on the one computer

	Type	Wibu - Auto Network
user defined name	Name	New dongle
use entry if Active is true	Active	true
not currently used	Filter	
number of times to retry	# retries	0
time to wait between retries	Wait time (seconds)	0.01

**CodeMeter - local:** a Dongle UNC is given as part of the information for this Type. The local computer is searched for a CodeMeter Container of the given Dongle UNC.

	Type	Codemeter - Local
user defined name	Name	New dongle
use entry if Active is true	Active	true
CodeMeter dongle UNC	Dongle UNC	
number of times to retry	# retries	0
time to wait between retries	Wait time (seconds)	0.01

**CodeMeter - auto local:** search the local computer for all CodeMeter Containers that have 12d Standalone dongles in them.

	Type	Codemeter - Auto Local
user defined name	Name	New dongle
use entry if Active is true	Active	true
not currently used	Filter	
number of times to retry	# retries	0
time to wait between retries	Wait time (seconds)	0.01

**CodeMeter - network:** a computer name is given as part of the information for this Type. The computer on the network with this name is searched for CodeMeter Containers with 12d network dongle numbers in them.

	Type	Codemeter - Network
user defined name	Name	New dongle
use entry if Active is true	Active	true
name of the computer	Server name	
number of times to retry	# retries	0
time to wait between retries	Wait time (seconds)	0.01

**CodeMeter - auto network:** search all computers on the network for CodeMeter Containers that have 12d network dongles in them.

	Type	Codemeter - Auto Network
user defined name →	Name	Codemeter Auto Network
use entry if Active is true →	Active	true
not currently used →	Filter	abc
number of times to retry →	# retries	0
time to wait between retries →	Wait time (seconds)	0.01

*Pause: wait for a given amount of time before going on to the next entry in the list*

	Type	Pause
user defined name →	Name	New dongle
use entry if Active is true →	Active	true
time to wait before going on to the next entry in the list →	Wait time (seconds)	0.01

- Delete** button  
*delete the highlighted entry.*
- Up Arrow** button  
*move the highlighted entry up in the list.*
- Down Arrow** button  
*move the highlighted entry down in the list.*
- Write** button

*write out the dongles.4d file. For more information on the **Write** button, go to the section [41.2.6 Writing Out Setup Files](#) in the Appendix [41 Setting Up & Configuring 12d](#).*

For information on the **dongles.4d** file installed with 12d Model, see [14.5.3.3.1 Dongles.4d Shipped with 12d Model](#).

**Important Note**

If the *dongles.4d* file being used by **12d Model** has no entries relating to network dongles, then any existing deprecated WIBU\_DONGLE\_4D and WIBU\_IPADDR environment variables for WIBU network dongles are still used.

Continue to the next section [14.5.3.4 Certify CodeMeter](#) or return to [14.5.3 Dongles](#).

### 14.5.3.3.1 Dongles.4d Shipped with 12d Model

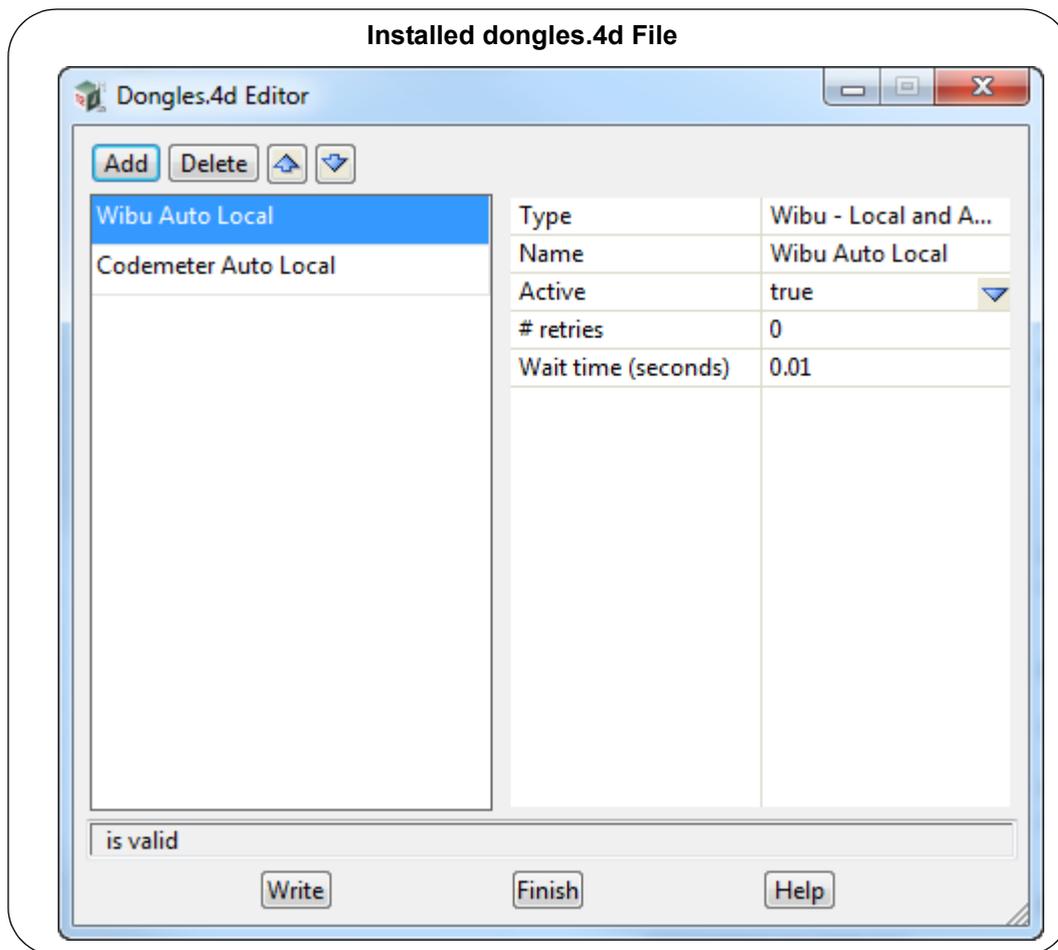
When 12d Model is installed, a **dongles.4d** file is included in **set\_ups**.

This shipped dongles.4d file only contains entries to as 12d Model to look for any local Wibu dongles and any local Codemeter Containers.

#### Important Note for Network Dongle Users

There are no entries for network dongles in the shipped dongles.4d file.

So if you are using Wibu network dongles or Codemeter Containers with network 12d dongles in them, then you need to have **your own** dongles.4d file that includes entries for network dongles. Your dongles.4d file is searched for as a Set Up file, or by using the environment variable DONGLES\_4D.



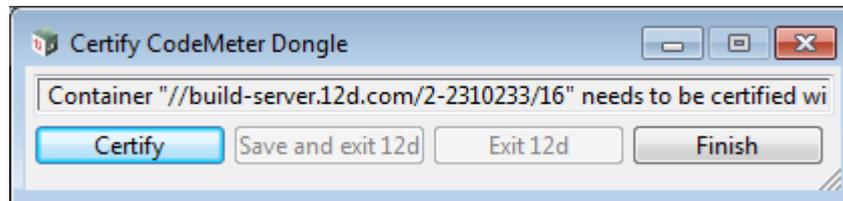
Return to [14.5.3.3 Dongles.4d Editor](#) or [14.5.3 Dongles](#).

### 14.5.3.4 Certify CodeMeter

**Position of option on menu:** Project =>Management =>Dongles => Certify CodeMeter

The **Certify CodeMeter** option is run when you are attached to the internet and checks that there is no problems with the **CodeMeter Container** that your **12d Model** license is coming from.

Selecting Certify CodeMeter brings up the **Certify CodeMeter Dongle** panel.



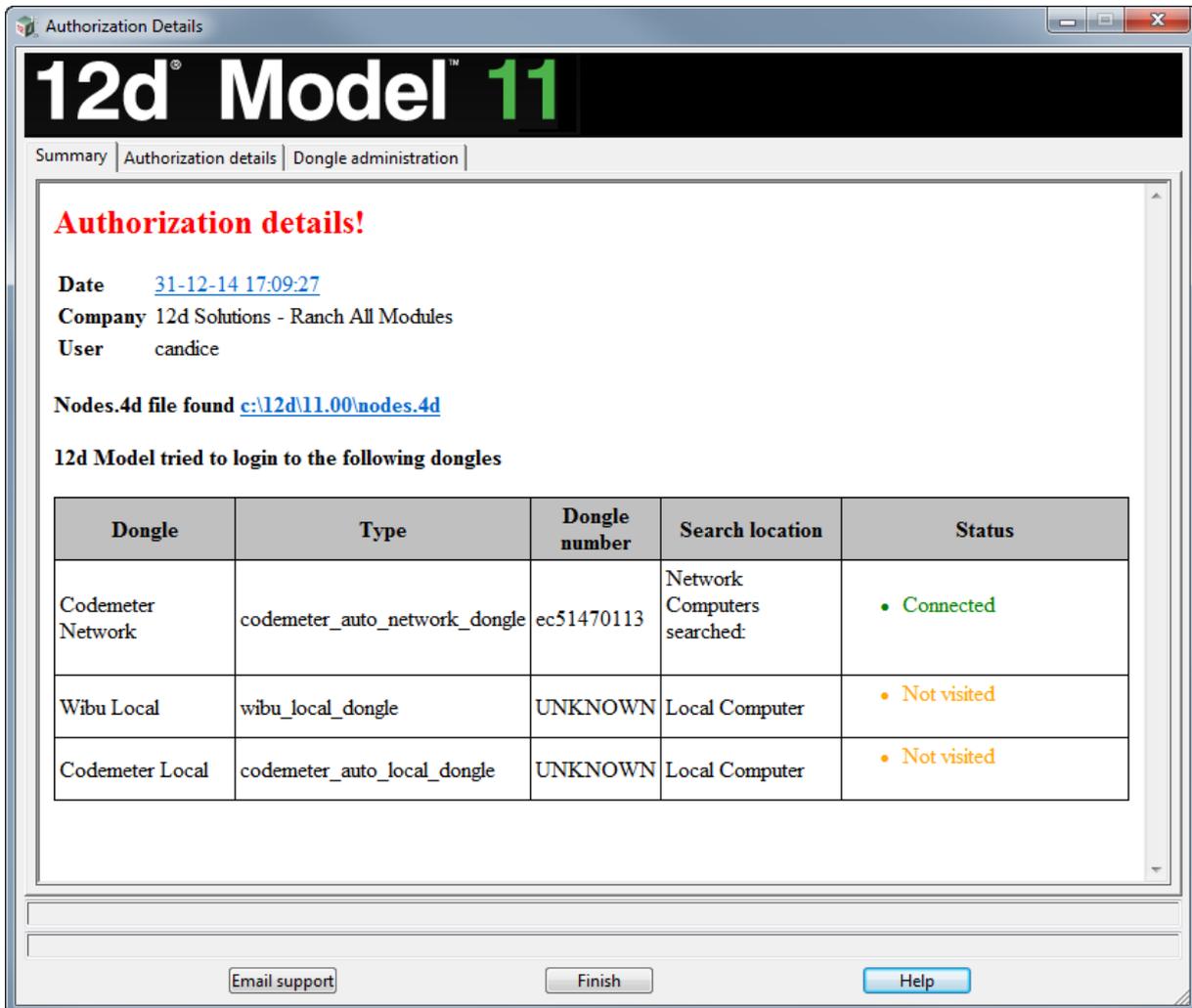
Continue to the next section [14.5.3.5 Authorization Details](#) or return to [14.5.3 Dongles](#).

### 14.5.3.5 Authorization Details

**Position of option on menu:** Project =>Management =>Dongles => Authorization details

The **Authorization Details** option shows similar information to the **Authorization Error** panel except in this case there is a match between a found 12d Model dongle and an entry in the nodes.4d file.

Selecting **Authorization details** brings up the **Authorization Details** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

#### Summary tab

*this summarises information from the **Authorization details** tab.*

*It shows the 12d Model dongle being used.*

#### Authorization details tab

*shows details about the*

- version of 12d Model*
- nodes.4d file used*
- dongles.4d file used*
- env.4d file used*
- Windows Environment*
- What is in the Output Window*

*Windows Registry*

*Windows Registry User*

### **Dongle administration tab**

*same as running the **Project =>Management =>Dongles =>Administration** option. See [14.5.3.2 Dongles Administration](#).*

### **Buttons at Bottom**

**Email support** button

*creates an email with an attachment of a zipped up copy of all the information on the **Authorization details** tab.*

For documentation on the **Authorization Error** panel, go to [14.5.3.5.1 Authorization Error](#).

Continue to the next section [14.5.3.6 Nodes.4d Editor](#) or return to [14.5.3 Dongles](#).

### 14.5.3.5.1 Authorization Error

This panel only appears if **12d Model** will not authorise for the selected project.

Before any selected **12d Model** project will open, a valid authorization is required.

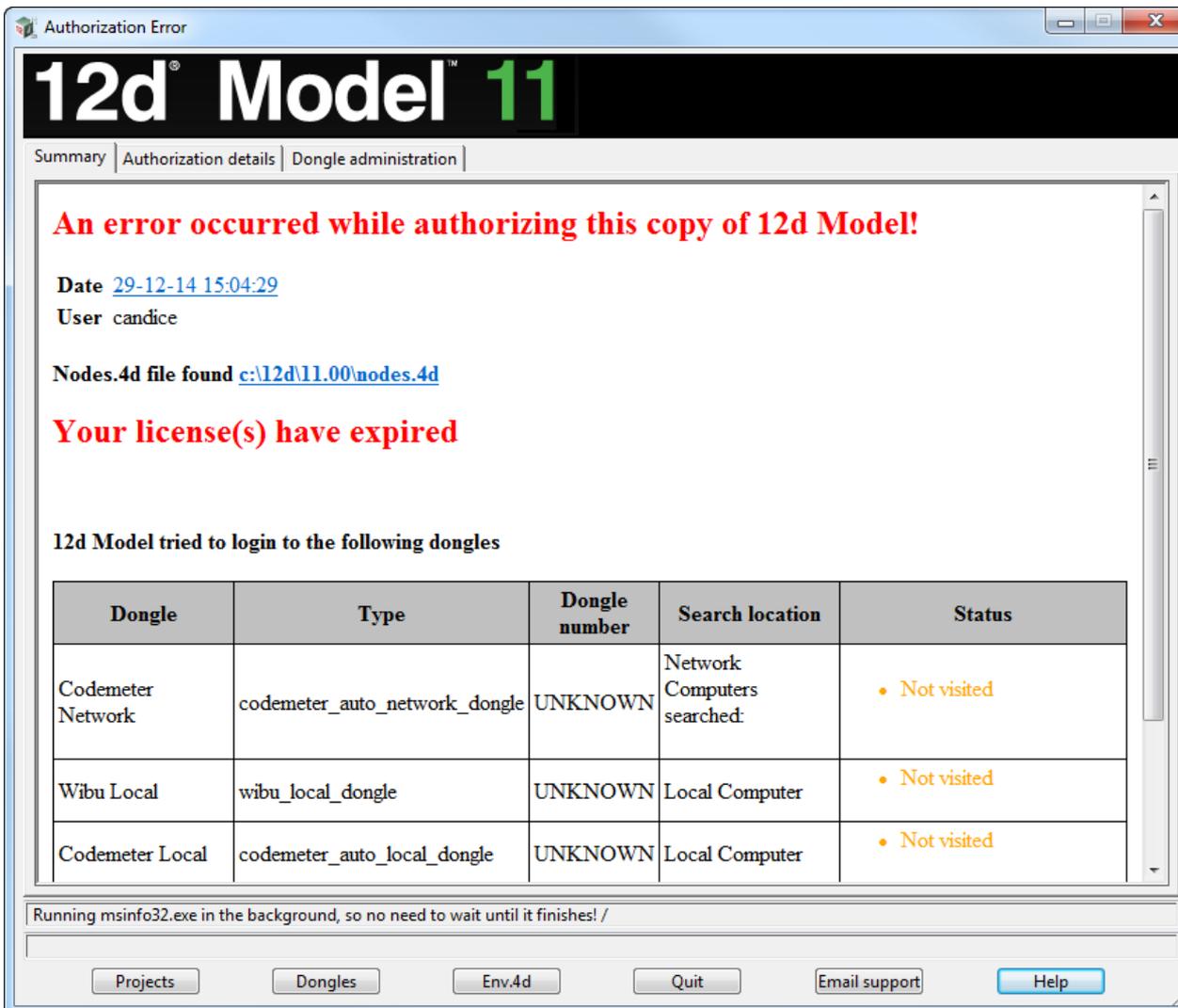
A valid authorization consists of:

1. a **12d Model** dongle number from a Wibu dongle or a CodeMeter Container
2. a valid entry in a nodes.4d to match the **12d Model** dongle number

If either of the above is missing, then instead of the project being opened, an **Authorization Error** panel is displayed with information to help find the error.

For more information on what is needing for authorising **12d Model**, see [14.5.3.1 12d Dongles, Dongles.4d, Nodes.4d, and Authorizing](#).

The **Authorization Error** panel contains information to help find out why there is an authorization error.



The fields and buttons used in this panel have the following functions.

Field Description                      Type                      Defaults                      Pop-Up

#### Summary tab

*this summarises information from the **Authorization details** tab.*

## Authorization details tab

*shows details about the*

*version of 12d Model  
nodes.4d file used  
dongles.4d file used  
env.4d file used  
Windows Environment  
What is in the Output Window  
Windows Registry  
Windows Registry User*

## Dongle administration tab

*same as running the **Project =>Management =>Dongles =>Administration** option. See [14.5.3.2 Dongles Administration](#).*

## Buttons at Bottom

**Projects** button

*opens the **12d Model** front screen so a **12d Model** project can be selected or created.*

**Dongles.4d** button

*opens the **Dongles.4d Editor** panel so that the *dongles.4d* file can be edited. See [14.5.3.3 Dongles.4d Editor](#).*

**Env.4d** button

*opens the **Env.4d Editor** panel so that the *env.4d* file can be edited.*

**Email support** button

*creates an email with an attachment of a zipped up copy of all the information on the **Authorization details** tab. This will provide useful information for someone trying to determine why the project is not authorising.*

Return to [14.5.3 Dongles](#).

### 14.5.3.6 Nodes.4d Editor

**Position of option on menu:** Project =>Management =>Dongles => Nodes.4d editor

The **nodes.4d** is an XML format and so can not be easily edited, or even viewed, in its native XML format.

There is a program shipped by **12d Solutions** that is used to install the nodes.4d file that can:

- (a) Install a **nodes.4d** file
- but also load an existing XML nodes.4d and
- (b) merge in another nodes file
- (c) look at all the entries in the file and display information on each entry (start & end dates, modules authorised etc)
- (d) move entries up and down
- (e) delete entries
- (f) create an html report on the entries
- (g) create a new **nodes** file from selected entries

This program is accessible from inside **12d Model** by

- (a) the **Nodes** button on the Front screen (the one before you select a project)
- (b) the option **Projects =>Management =>Dongles =>Nodes.4d editor**

and also from outside **12d Model**

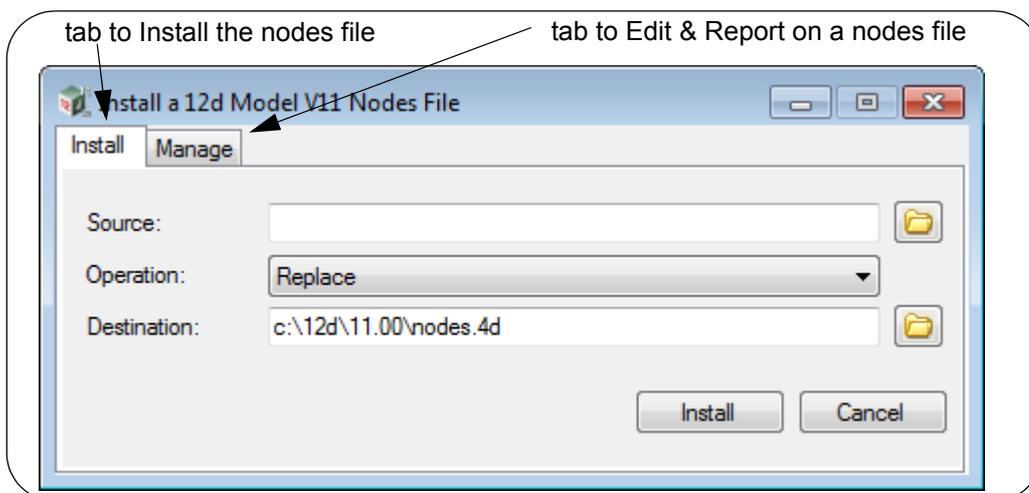
- (c) the program is called 12dNodesUtility.exe and is installed as a 32-bit program in

**Program Files (x86) \12d\Nodes\11.0**

The **Nodes.4d Editor** option is for editing the **Nodes.4d** file.

Note that program is actually external to **12d Model** and when it is run, it is running independently of **12d Model**.

Selecting **Nodes.4d editor** brings up the external program and its panel **Installing a 12d Model 11 Nodes File**:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Install** **tab**

*this tab is for installing the nodes.4d file.*

**Source** file box file browser

*the full path name of the nodes file to install.*

**Operation** choice box Replace Replace, Append, Prepend

*if **Replace**, the Destination file is replaced by the Source file.*

*If **Append**, the Source file is added to the end of the Destination file.*

*If **Prepend**, the Source file is added to the beginning of the Destination file.*

**Destination** file box file browser

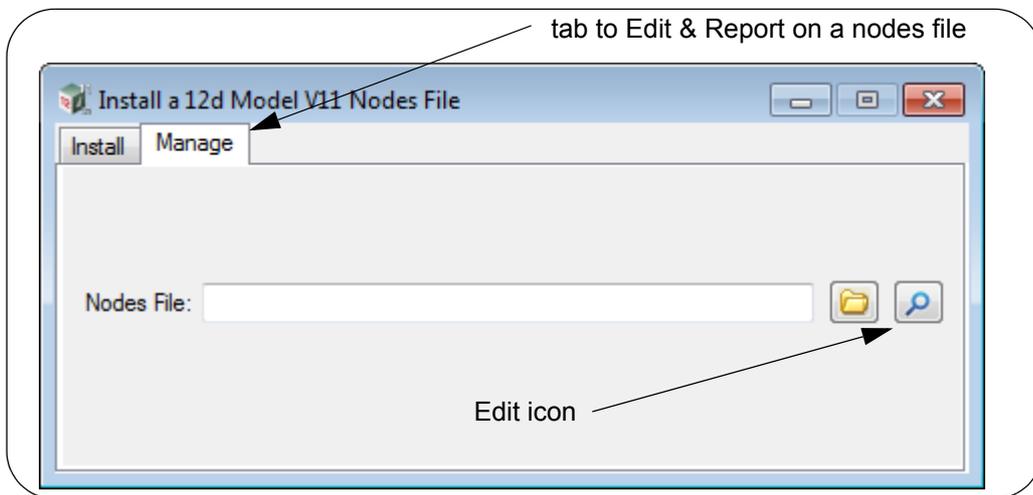
*the full path name of where to install the **nodes.4d** file.*

**Install** button

*install the nodes.4d file.*

**Manage** **tab**

*this tab is for editing and reporting a nodes.4d file.*



**Nodes file** file box file browser

*the full path name of the nodes file to edit/modify/report.*

*After the nodes.4d file has been fill in, click on the Edit icon (Looking glass icon) to bring up the **Node Contents** panel.*

*The panel shows all the Client Entries in the nodes.4d file in a tree structure with all the dongle records for a particular Client Entry displayed when the + is clicked on in front of the Client Entry.*

**Note:** *The Client Entry contains all the dongles records that were produced for a Client at the one time. These dongle records can only stay inside that one Client Entry, even though there may be other Client Entries of the same name.*

*Dongle Records can be moved up or down within a Client Entry but can't be moved to a different Client Entry.*

*A Client Entry can be moved up and down in the Tree.*

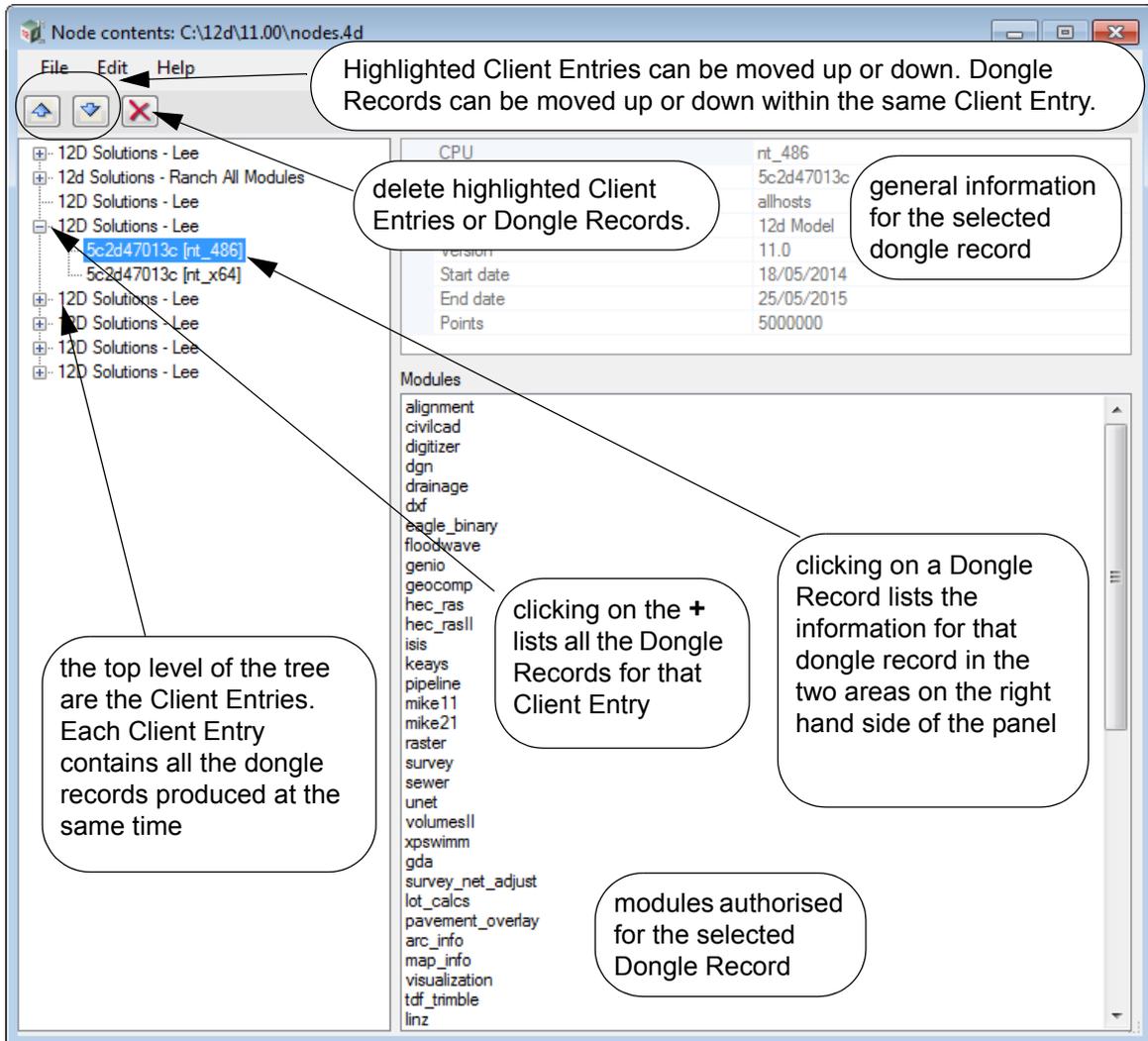
*Highlighted Client Entries can be deleted and all the Dongle Records for the Client Entry will also be deleted.*

*Highlighted Dongle Records can be deleted.*

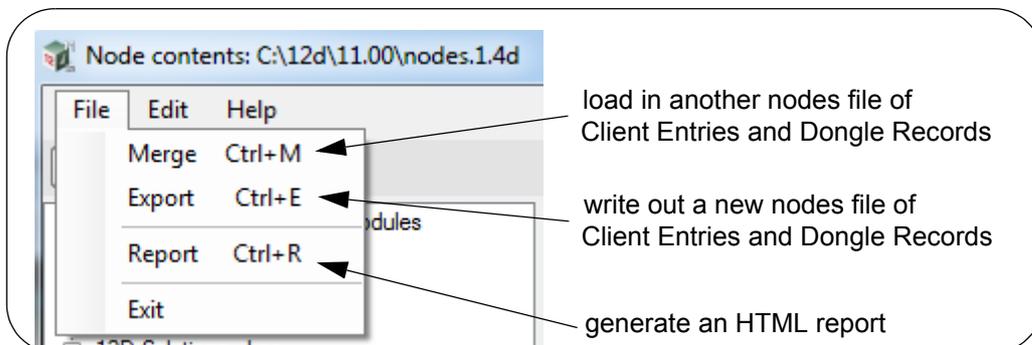
**File** *has options to produce a HTML report and write out to a new nodes file sa subset of the Client*

Entries and Dongle Records.

Clicking on X removes the **Node Contents** panel

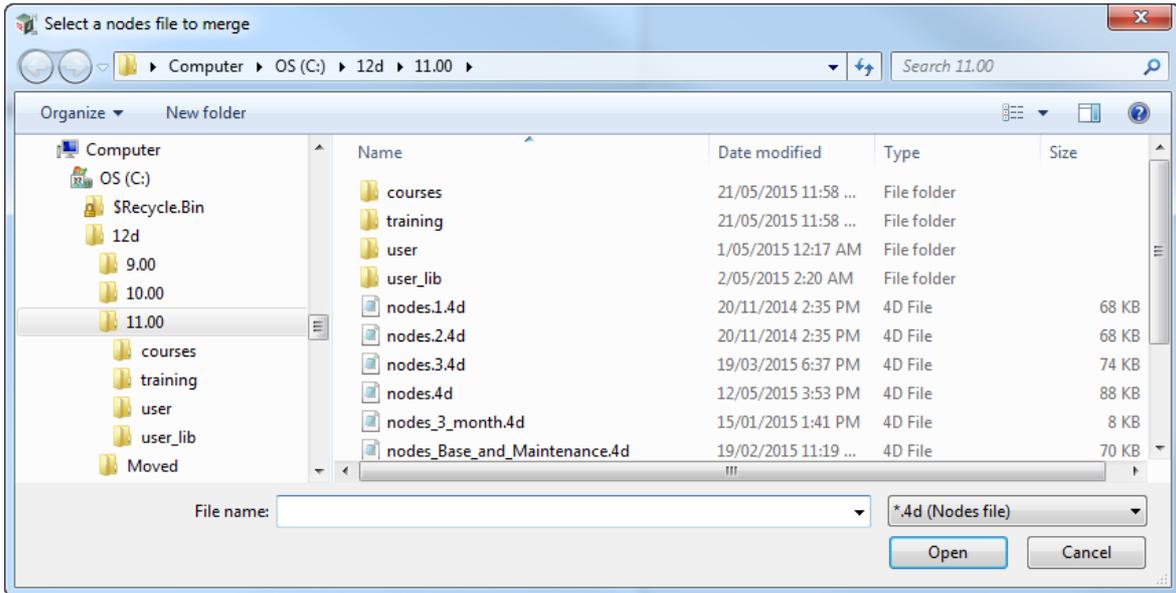


Clicking on File brings up the



**1. Merge** menu item

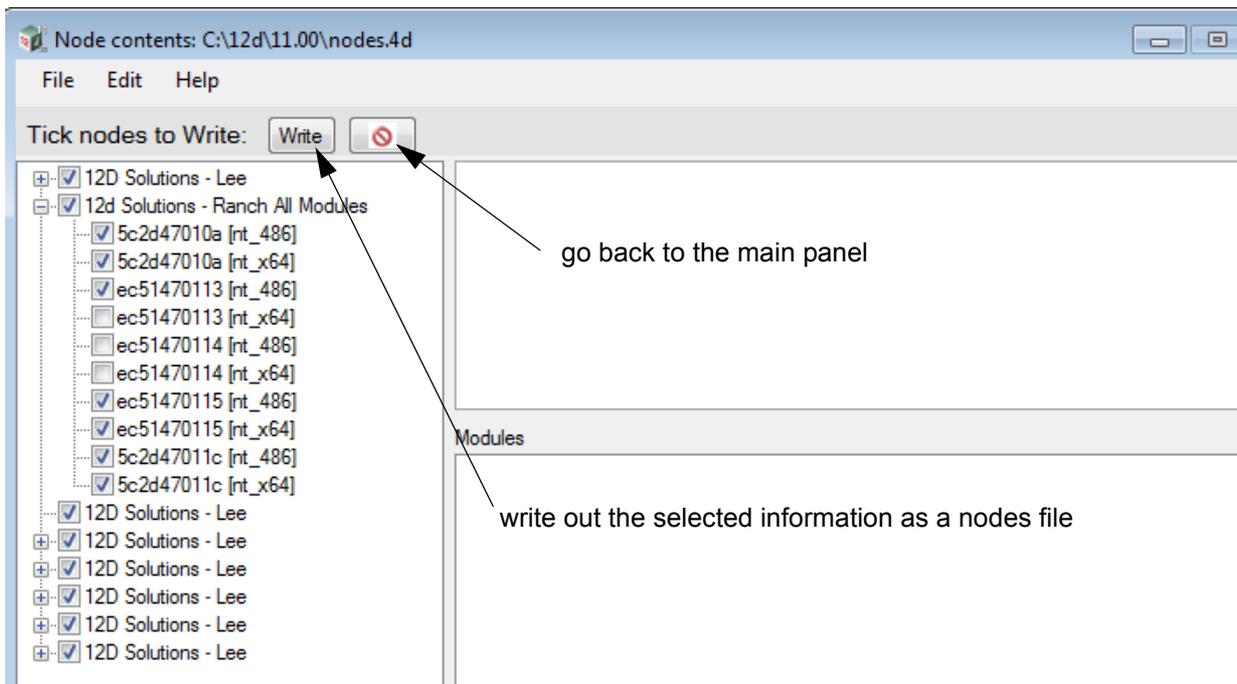
clicking on *Merge* brings up the Microsoft browser to obtain the another nodes.4d read in.



The new **Client Entries** and **Dongle Records** are added to the bottom of the list of existing **Client Entries** ready for editing.

**2. Export** menu item

clicking on **Export** brings a Tree showing all the **Client Entries** and **Dongle Records** but with a tick box to indicate if the information is to be written out or not, to the new nodes file.



**Write** button

clicking on **Write** brings up the Microsoft **Save As** browser to obtain the name of the file to write the selected information out to as a node.4d file.

**Stop** button

clicking on **Stop** takes you back to the **Node Contents** panel

**3. Report** menu item

*clicking on **Report** brings up the Microsoft **Save As** browser to obtain the name of the file to write the HTML report to.*

*The report will list all the **Client Entries**. And for each **Client Entry**, the **Dongle Records** in it showing the modules authorised for that Dongle Record in blue.*

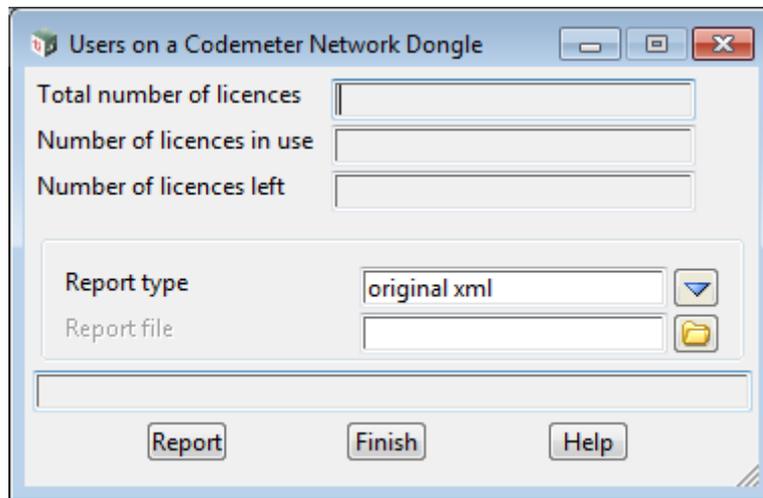
Continue to the next section [14.5.3.7 CodeMeter Network User](#) or return to [14.5.3 Dongles](#).

### 14.5.3.7 CodeMeter Network User

**Position of option on menu:** Project =>Management =>Dongles => CodeMeter network users

This option shows the number of **12d Model** licenses on the CodeMeter network dongles and how many are used/unused. A report can also be produced showing details for each individual network dongle.

Selecting CodeMeter network users brings up s panel **Users on a CodeMeter Network Dongle**:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Total number of licenses** output only

*the total number of 12d Model licenses on the CodeMeter network dongles. This number is updated when **Report** button is pressed.*

**Number of licenses in use** output only

*the number of 12d Model licenses currently being used.*

**Number of licenses left** output only

*the number of 12d Model licenses left to be used.*

**Report type** choice box html report, original xml, <customize>

*output format for the report. An XML file will be produced and then if required, converted to the selected report.*

*For information on setting up custom reports from the generated XML file using xslts, see [4.30 Setting Up XML Reports](#).*

**Report file** file box

*if **not blank**, an XML file will be created and a report of this name, and of the type given by **Report type** will be generated from the XML file.*

*If **blank**, no report is created but the number of licenses information is still written to the appropriate panel fields.*

**Report** button

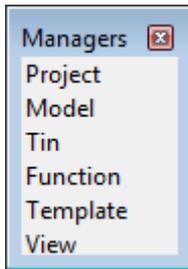
*check how many licenses are on CodeMEter network dongles and write the values to the panel fields. If Report file is not blank then a report is also produced.*

## 14.5.4 Managers

**Position of menu:** Project =>Management =>Managers

The **Managers** display in a tree structure information and settings for all the models, tins, functions and templates both in the project and removed from the project, and for all the views in the project.

The **Managers** walk-right menu contains a manager for the entire project and separate managers for modes, tins, functions templates and views.



For the option:

*Project*, go to [14.5.4.1 Project Manager](#).

*Model*        [14.5.4.2 Model Manager](#).

*Tin*            [14.5.4.3 Tin Manager](#).

*Function*     [14.5.4.4 Function Manager](#).

*Template*      [14.5.4.5 Template Manager](#).

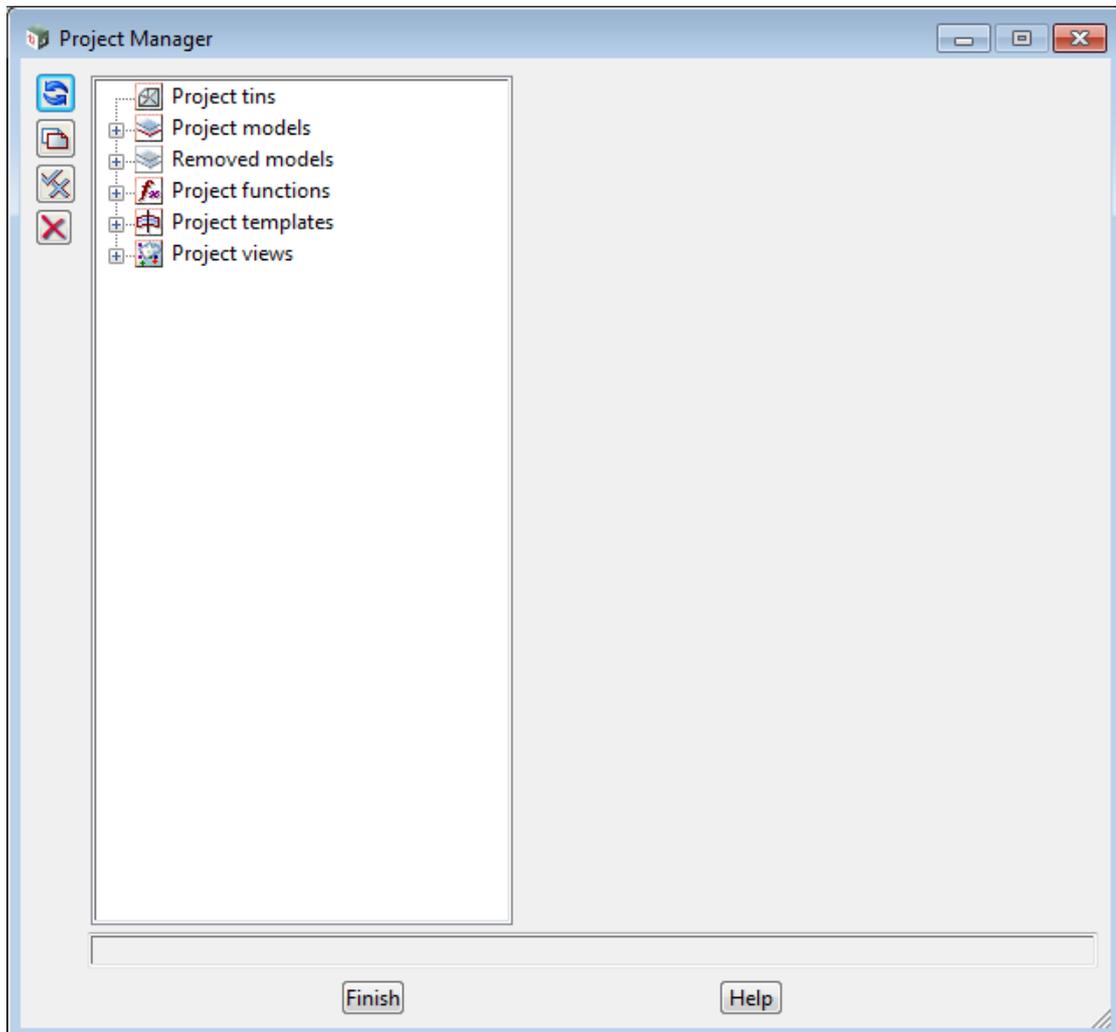
*View*          [14.5.4.6 View Manager](#).

### 14.5.4.1 Project Manager

**Position of option on menu:** Project =>Management =>Managers =>Project

The **Project Manger** displays in a tree structure nodes for all the models, tins, functions and templates both in the project and removed from the project, and all the views in the project.

Selecting Project displays the **Project Manager** panel.



If there are any models/tins/functions/templates in the Project, then there will be a **Project models/tins/functions/templates** node in the tree and that node has sub nodes for all the models/tins/functions/templates in the project. The name of the sub node is the name of the model/tin/function/template.

If there are any models/tins/functions/templates that have been removed from the Project, then there will be a **Removed models/tins/functions/templates/views** node in the tree and that node has sub nodes for all the removed models/tins/functions/templates. The name of the sub node is the name of the removed model/tin/function/template

If there are any views in the Project, then there will be a **Project view** node in the tree with the views as sub nodes. The name of sub node is the name of the view.

The information and setting displayed for these nodes is the same as those in the separate Model/Tin/Function/Template/View Managers and so will be documented in those sections.

See

[14.5.4.2 Model Manager](#)

[14.5.4.3 Tin Manager](#)

[14.5.4.4 Function Manager](#)

[14.5.4.5 Template Manager](#)

[14.5.4.6 View Manager](#)

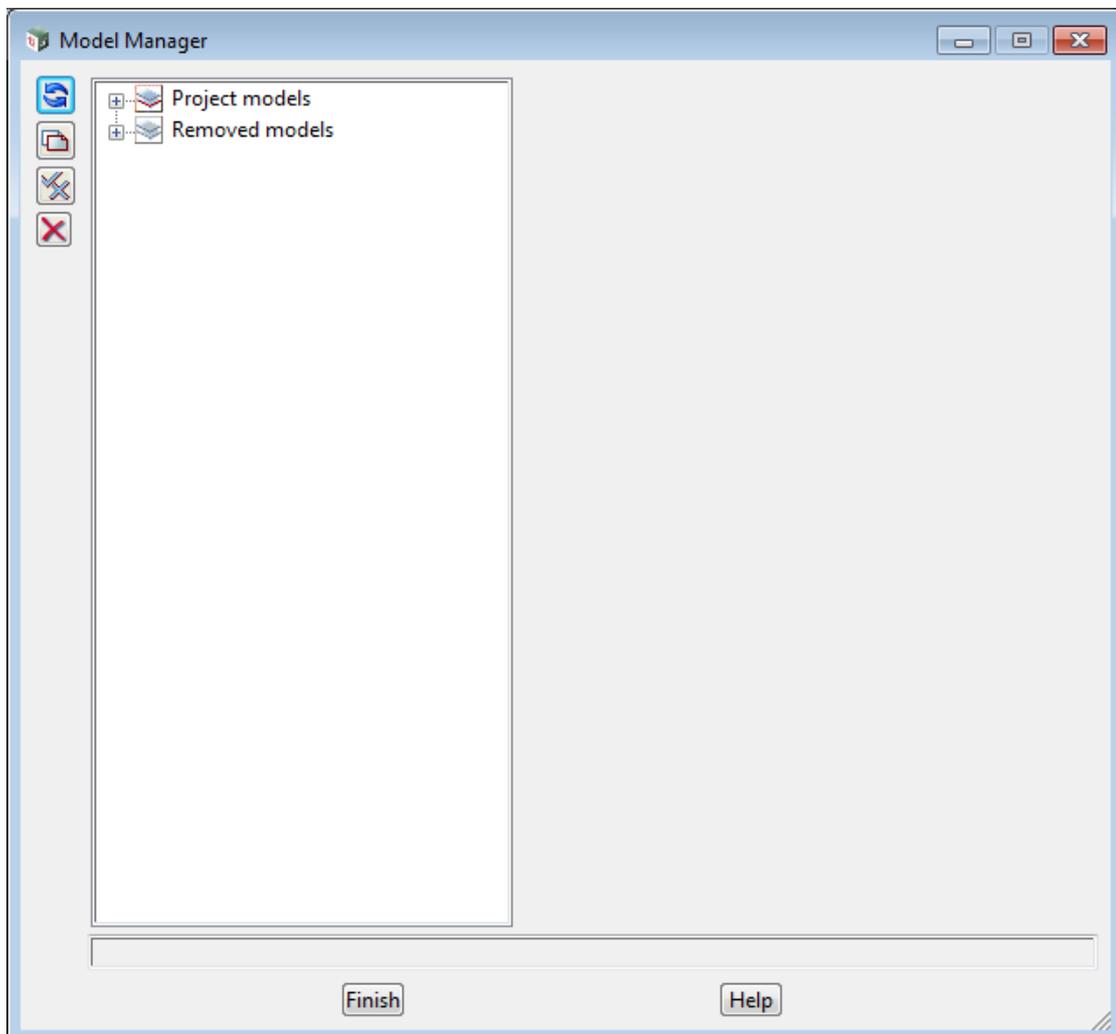
Continue to the next section [14.5.4.2 Model Manager](#) or return to [14.5.4 Managers](#).

## 14.5.4.2 Model Manager

**Position of option on menu:** Project =>Management =>Managers =>Model

The **Model Manger** displays in a tree structure all the settings for models in the project, and the names of all the removed models.

Selecting **Model** displays the **Model Manager** panel.



If there are any models in the Project, then there will be a **Project models** node in the tree.

Expanding the **Project models** node lists all the models in the project as sub nodes, and expanding the sub node of a project model displays information and settings for the model on the right hand side of the panel. The name of the sub node is the name of the model.

Many of the settings can then be changed, and the changes take place immediately without having to press any other button.

If there are any models that have been removed from the Project, then there will be a **Removed models** node in the tree.

Expanding the **Removed models** node lists all the models removed from the project as sub nodes but no other information or settings can be displayed for removed models. The name of the sub node is the name of the removed model.

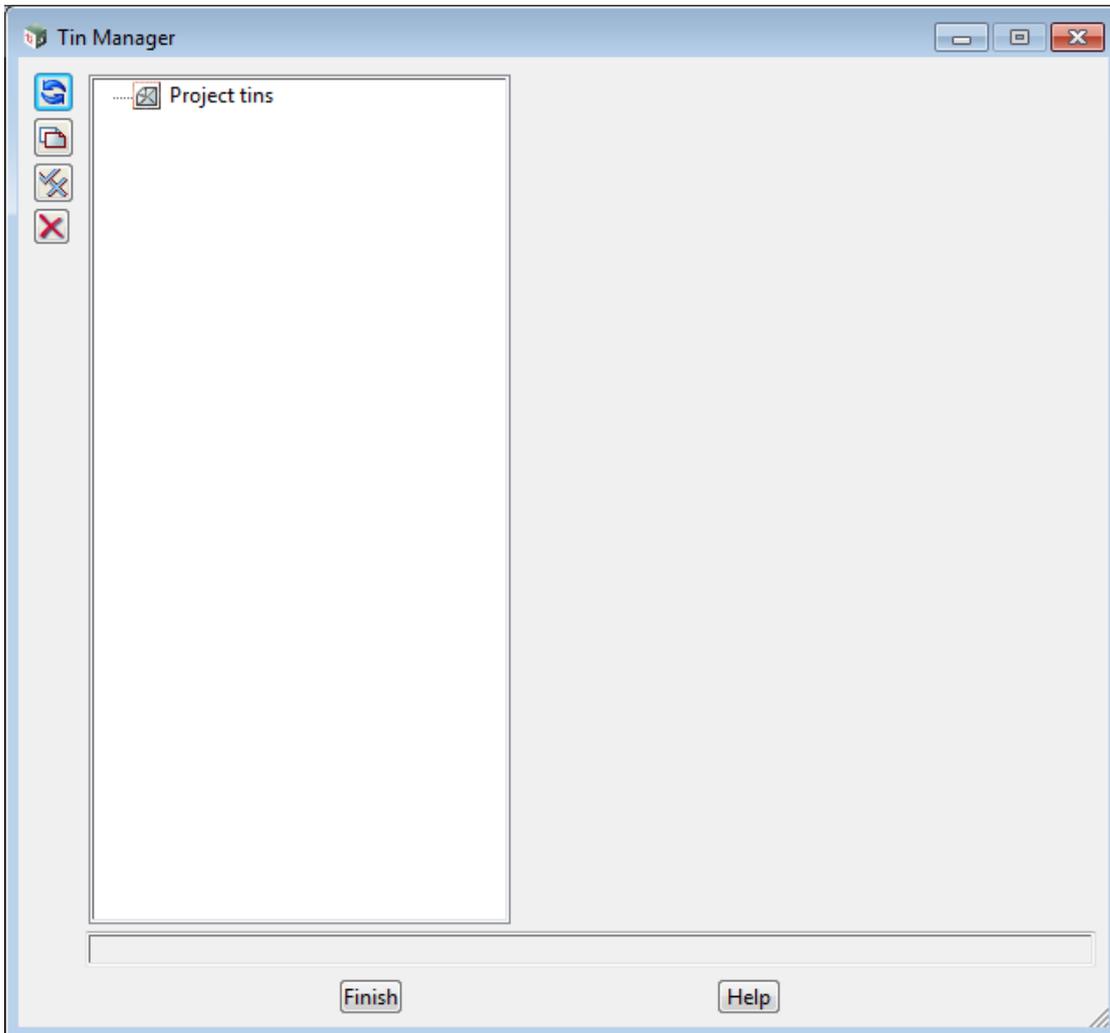
Continue to the next section [14.5.4.3 Tin Manager](#) or return to [14.5.4 Managers](#).

### 14.5.4.3 Tin Manager

**Position of option on menu:** Project =>Management =>Managers =>Tin

The **Tin Manger** displays in a tree structure, all the settings for tins in the project, and the names of all the removed tins.

Selecting **Tin** displays the **Tin Manager** panel.



If there are any tins in the Project, then there will be a **Project tins** node in the tree.

Expanding the **Project tins** node lists all the tins in the project as sub nodes, and expanding the sub node of a project tin displays information and settings for the tin on the right hand side of the panel. The name of the sub node is the name of the tin.

Many of the settings can then be changed, and the changes take place immediately without having to press any other button.

If there are any tins that have been removed from the Project, then there will be a **Removed tins** node in the tree.

Expanding the **Removed tins** node lists all the tins removed from the project as sub nodes but no other information or settings can be displayed for removed tins. The name of the sub node is the name of the removed tin.

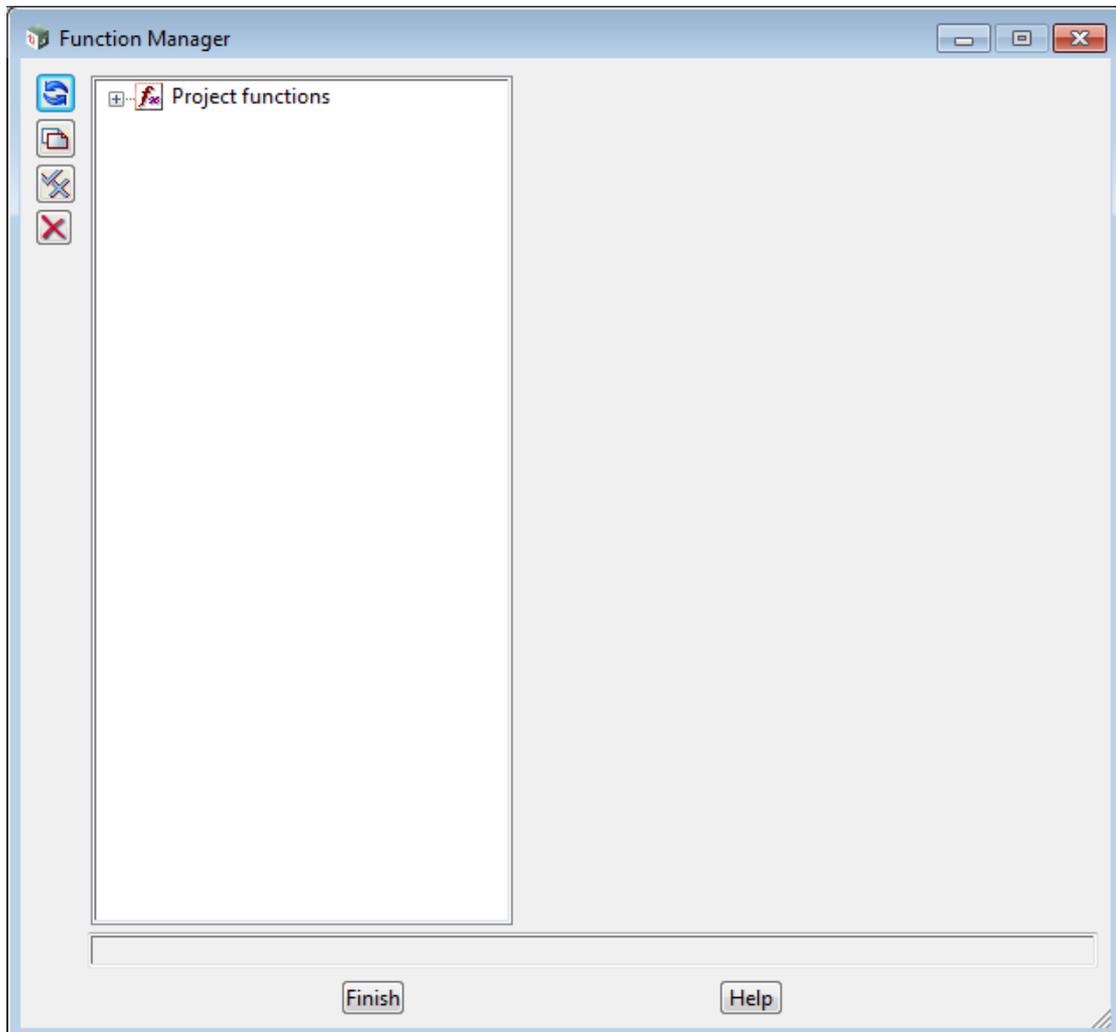
Continue to the next section [14.5.4.4 Function Manager](#) or return to [14.5.4 Managers](#).

## 14.5.4.4 Function Manager

**Position of option on menu:** Project =>Management =>Managers =>Function

The **Function Manger** displays in a tree structure, all the settings for functions in the project, and the names of all the removed functions.

Selecting **Function** displays the **Function Manager** panel.



If there are any functions in the Project, then there will be a **Project functions** node in the tree.

Expanding the **Project functions** node lists all the functions in the project as sub nodes, and expanding the sub node of a project function displays information and settings for the function on the right hand side of the panel. The name of the sub node is the name of the function.

Many of the settings can then be changed, and the changes take place immediately without having to press any other button.

If there are any functions that have been removed from the Project, then there will be a **Removed functions** node in the tree.

Expanding the **Removed functions** node lists all the functions removed from the project as sub nodes but no other information or settings can be displayed for removed functions. The name of the sub node is the name of the removed function.

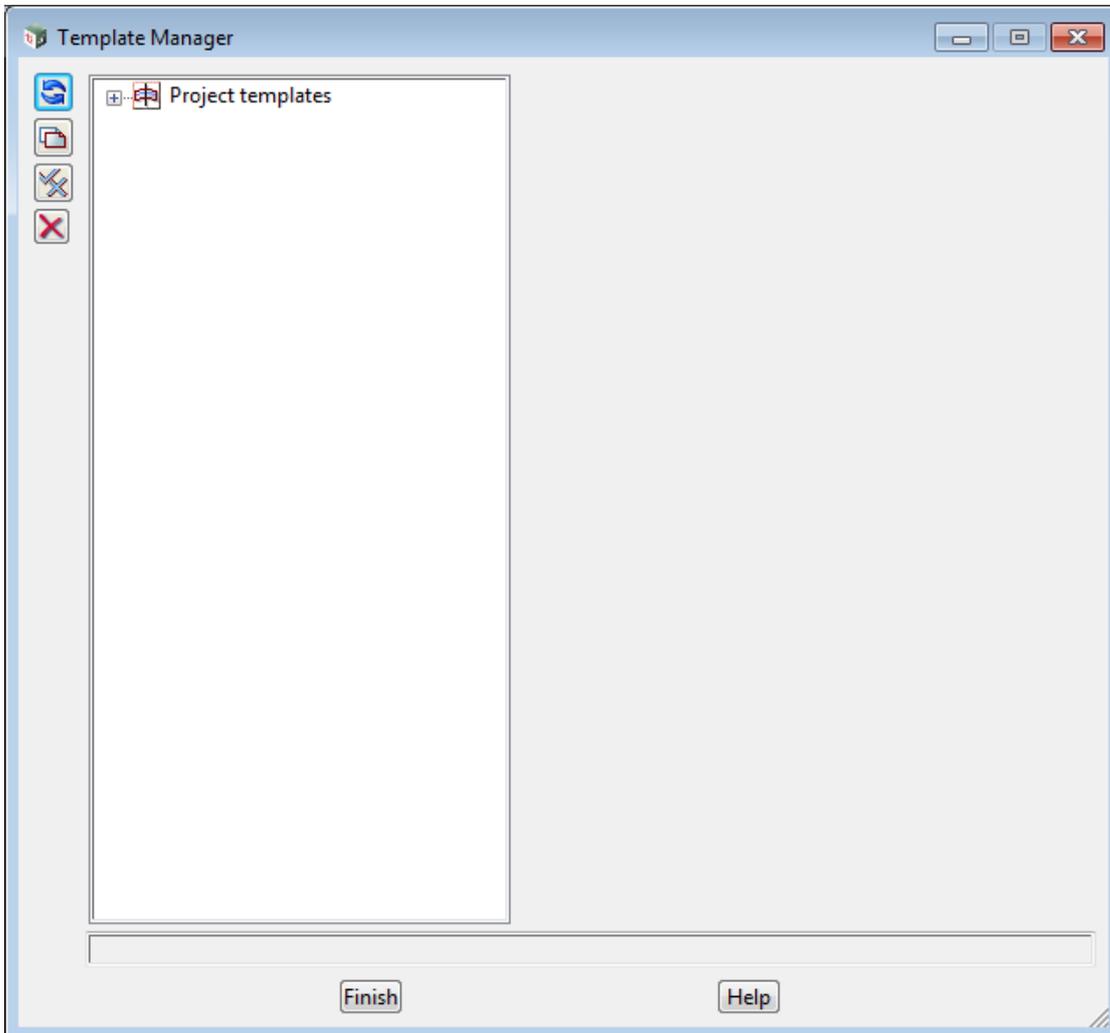
Continue to the next section [14.5.4.5 Template Manager](#) or return to [14.5.4 Managers](#).

## 14.5.4.5 Template Manager

**Position of option on menu:** Project =>Management =>Managers =>Template

The **Template Manger** displays in a tree structure, all the settings for templates in the project, and the names of all the removed templates.

Selecting **Template** displays the **Template Manager** panel.



If there are any templates in the Project, then there will be a **Project templates** node in the tree.

Expanding the **Project templates** node lists all the templates in the project as sub nodes, and expanding the sub node of a project template displays information and settings for the template on the right hand side of the panel. The name of the sub node is the name of the template.

Many of the settings can then be changed, and the changes take place immediately without having to press any other button.

If there are any templates that have been removed from the Project, then there will be a **Removed templates** node in the tree.

Expanding the **Removed templates** node lists all the templates removed from the project as sub nodes but no other information or settings can be displayed for removed templates. The name of the sub node is the name of the removed template.

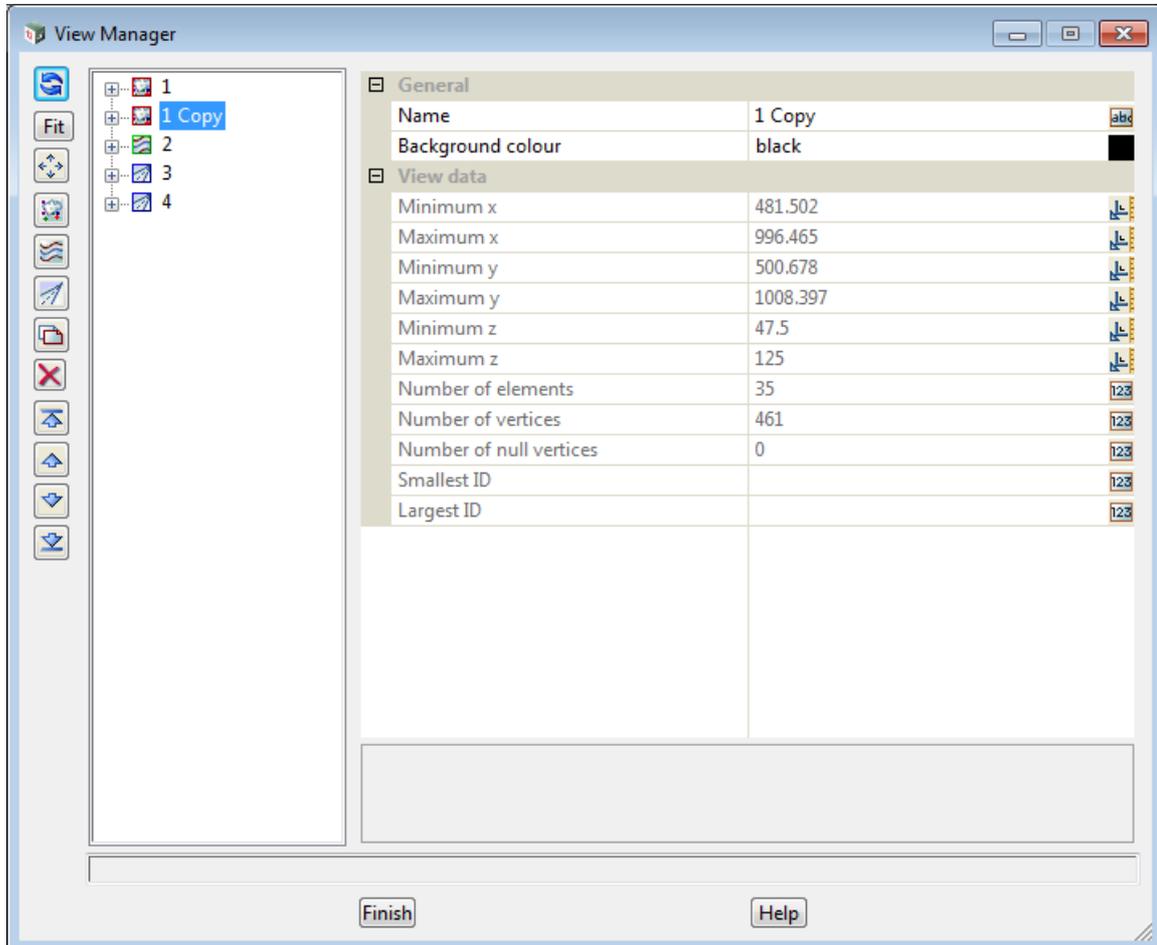
Continue to the next section [14.5.4.6 View Manager](#) or return to [14.5.4 Managers](#).

### 14.5.4.6 View Manager

**Position of option on menu:** Project =>Management =>Managers =>View

The **View Manger** displays in a tree structure, all the settings for views in the project.

Selecting **View** displays the **View Manager** panel.



If there are any views in the Project, then there will be a node (**view node**) in the tree with the **name of the view** as the **node name**.

Expanding a **view node** lists information and setting for that view as sub nodes, and displays values for the nodes and sub nodes in the right hand side of the panel.

Once displayed in the right had side, many of the settings can be changed, and the changes will take place immediately without having to press another button.

For each **view node**, the sub nodes, settings and information displayed is the same as for the **View Properties** panels.

## 14.5.5 Toggle Topmost Buttons

**Position of option on menu:** Project =>Management =>Toggle topmost buttons

Toggles on buttons to use instead of a keyboard or mouse when working on a tablets.

Clicking on **Toggle topmost buttons** toggles the following buttons on and off.



# 15. File Menu

Note that the menu **File I/O** has been changed to **File** on the Main Menu.

This is so that the menu will fit across the screen on some tablets, especially when using a larger font.

File on Main menu

create floating File I/O menu

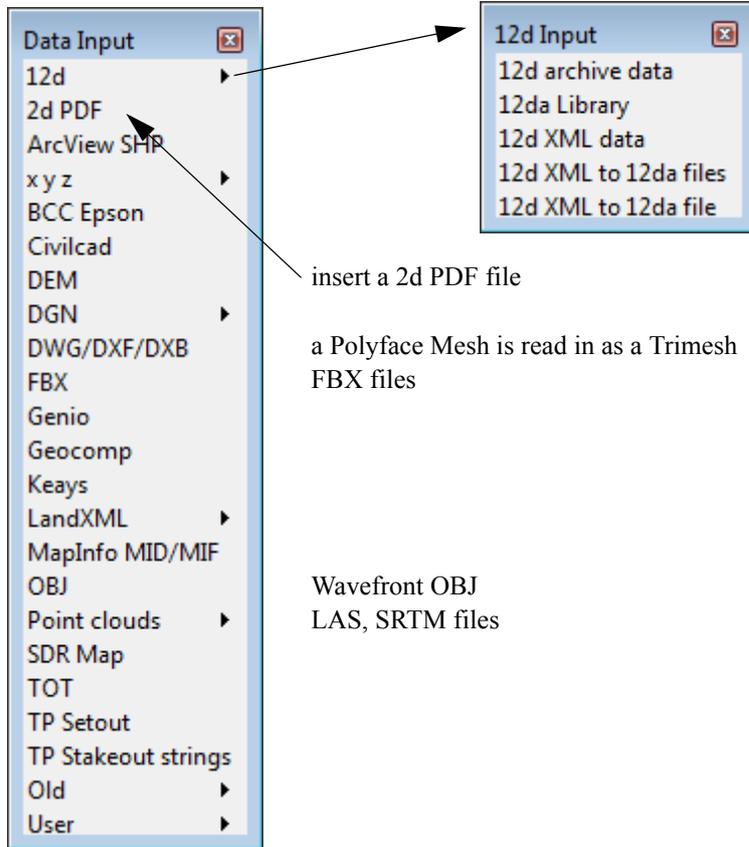
Data input	read in data files
Data output	write out data files
Layouts	read/write screen layout files
ADAC	create, read and write ADAC XML files
Digitizer	digitizing option
GIS	GIS interface
Label Map files	read/write Label Map Files
Map files	read/write Map Files
Range files	read/write range files
Screen dump	screen dump of window
Templates	read/write design templates
Textstyle input	read in textstyle definitions file
Edit file	edit a file
User	File I/O User menu

See

- [15.1 Data Input](#)
- [15.2 Data Output](#)
- [27 ADAC](#)
- [15.3 Map Files =>Apply](#)
- [15.4 Range File Menu](#)

# 15.1 Data Input

Position of menu: File I/O =>Data input



**12d Model** archive file  
insert a 12d archive file  
read 12d XML file  
convert 12d XML files to 12da files  
convert 12d XML files to a 12da file

insert a 2d PDF file

a Polyface Mesh is read in as a Trimesh  
FBX files

Wavefront OBJ  
LAS, SRTM files

See

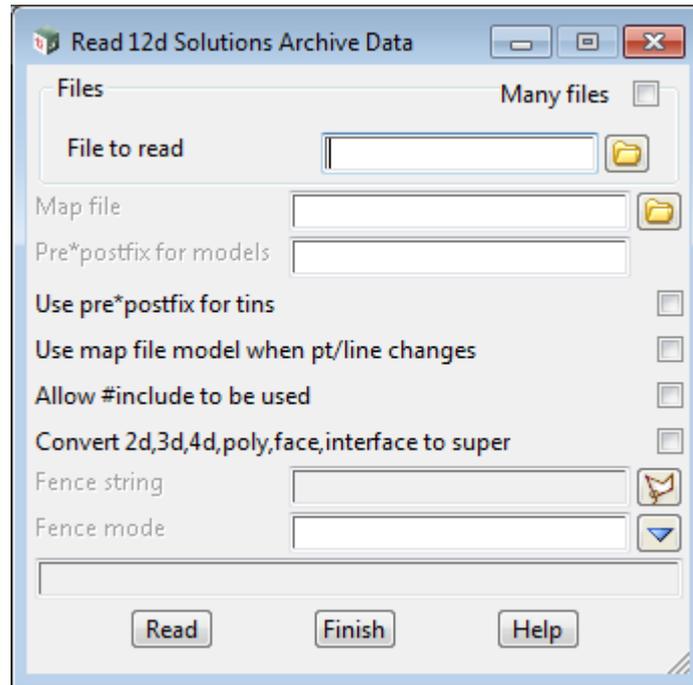
- [15.1.1 12d Archive Input](#)
- [15.1.2 Read 12d XML Data](#)
- [15.1.3 12d XML to 12da files](#)
- [15.1.4 12d XML to 12da file](#)
- [15.1.5 DWG PolyFace Mesh to Trimesh](#)
- [17.6.3 Trimeshes from FBX File](#)
- [15.1.6 Import 2D PDF](#)
- [15.1.7 Wavefront OBJ Input](#)
- [15.1.8 Input Point Cloud Files](#)

## 15.1.1 12d Archive Input

Position of option on menu: **File =>Data input =>12d =>12d archive data**

The **12d Archive** format is a special format defined by 12d Solutions to allow data to be easily transferred from other programs into 12d Solutions software such as **12d Model**. The **12d Archive** format is given in the Appendix.

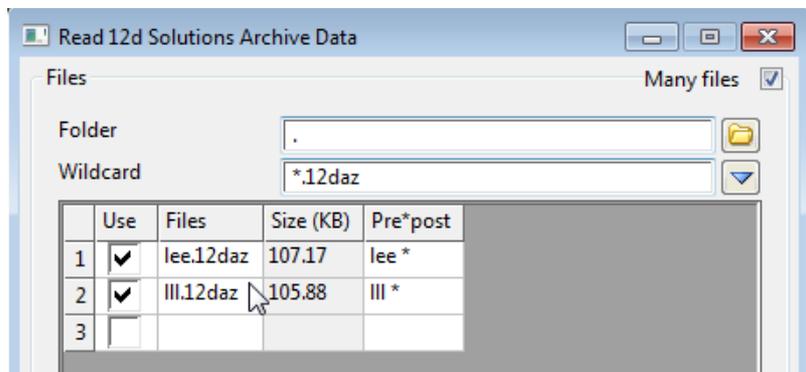
Selecting 12d archive data brings up the **Read 12d Solutions Archive Data** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Many files</b>	tick box		

*if **ticked**, a grid to allow multiple 12d archive files to be read in, is opened. A wild card is used to select all the files to be read in.*



<b>Folder</b>	folder box
	<i>folder to search for files using the Wild card</i>
<b>Wildcard</b>	text box
	<i>wild card to use in search for files in the given folder</i>
<b>Use</b>	tick box

if *ticked*, read in the file

**Files** file box

name of the file in the folder

**Size (KB)** output only

file size

**Pre\*post** text box

if **non blank**, pre\*post text to use for the models in this 12d archive file.

If **blank**, use the pre\*post text from the Pre\*postfix for models panel field.

*Note* - if a non-blank value for **Pre\*post** is given in the column for a file then the **Pre\*postfix for models** is ignored.

**File to read** file box \*.12daz, \*.12da or \*.4da files

name of the **12d Model** Archive file to be read in.

*Note* that **Drag and Drop** works for 12daz, 12da and 4da files.

It will also work for files that are email attachments in Outlook 2002 and above.

If you are dragging and dropping more than one **12daz, 12da or 4da** file at a time, then one panel will be opened and all the dropped files listed in the **Many Files** grid.

**Map file** file box \*.mapfile, \*.mf files

if **not blank**, the name of the 12d Map File to be used for all strings read in, including any files given with the **Many files** mode ticked on.

If **blank**, no map file is used

*When using a map file, the string name is used as the entity-name for matching with the keys in the map file.*

**Pre\*postfix for models** pre\*postfix box

if non-blank, a prefix and a postfix to be applied to the model names used in the map file.

*Note* - if a non-blank value for **Pre\*post** is given in the column for a file then the **Pre\*postfix for models** is ignored.

**Use pre\*postfix for tins** tick box not ticked

if ticked, a prefix and a postfix are to be applied to any tin names in the 12d Archive data.

**Warning** - if a tin already exists in 12d Model with the tin name, then the tin cannot be read in from the 12d Archive file.

**Use map file model when pt/line changes** tick box

if **not ticked** and the pt/line type of the string does not match that in the map file, then the string is placed in.

If tick, the.

**Allow #include to be used** tick box

if ticked, expand an files referenced by an #include.

If not ticked, ignore the #includes

**Convert 2d, 3d, 4d, poly, face, interface to super** tick box

if ticked, non super string versions of 2d/3d/4d/poly/face/interface strings are converted to super strings.

**Fence string** polygon box

A polygon is selected/created and used with the Fence mode choice to restrict the selection of strings read in from the 12d Archive file.

Clicking LB on the Select Polygon icon on the right hand side of the Fence string field allows the user to select a polygon.

Clicking RB on the Select Polygon icon on the right hand side of the Fence string field brings up the **Polygon Choice** Box for the user to select a method of creating/selecting a polygon.



*Clicking MB does nothing.*

**Fence mode**                      choice box                      available choices

*The strings selected are then restricted to those using the polygon and satisfying the Fence mode:*

Select Choice	
String inside	String inside - strings totally inside the polygon
String outside	String outside - strings totally outside the polygon
String crossing	String crossing - strings crossing the polygon
String inside/crossing	String inside/crossing - strings totally inside or crossing the polygon
String outside/crossing	String outside/crossing - strings totally outside or crossing the polygon

**Fence Modes**

**Read**                                      button

*reads the data in.*

**Special Note**

Pressing the <Esc> key will interrupt and terminate the reading in of the files.

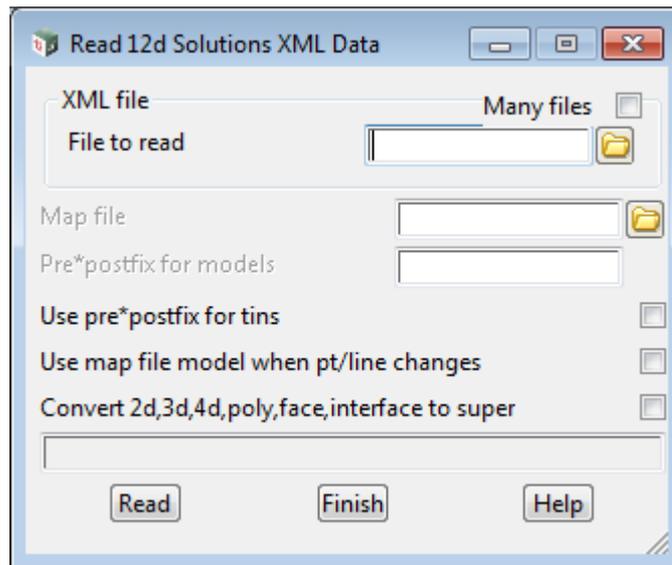
## 15.1.2 Read 12d XML Data

**Position of option on menu:** File I/O =>Data input =>12d =>12d XML data

The **12d XML** format is a special format defined by 12d Solutions to allow data to be easily transferred from other programs into 12d Solutions software such as **12d Model**. The **12d XML** format is given in the Appendix.

**Note** - 12d XML contains the same information as a 12da file but it is a XML format.

Selecting 12d XML data brings up the **Read 12d Solutions XML Data** panel:

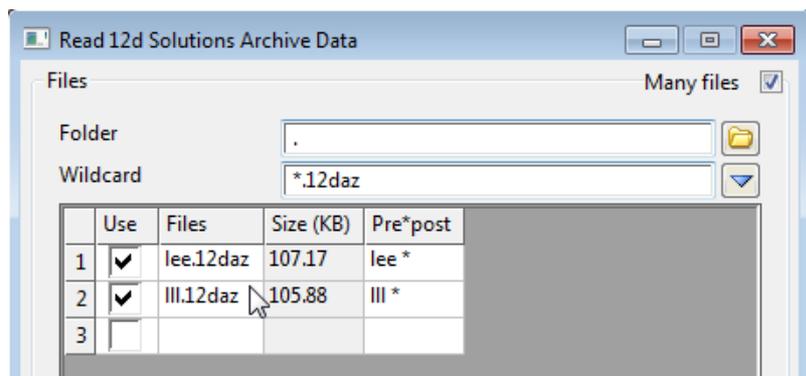


The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Many files</b>	tick box		
-------------------	----------	--	--

*if **ticked**, a grid to allow multiple 12d XML files to be read in, is opened. A wild card is used to select all the files to be read in.*



<b>Folder</b>	folder box
---------------	------------

*folder to search for files using the Wild card*

<b>Wildcard</b>	text box
-----------------	----------

*wild card to use in search for files in the given folder*

<b>Use</b>	tick box
------------	----------

*if **ticked**, read in the file*

**Files** file box

*name of the file in the folder*

**Size (KB)** output only

*file size*

**Pre\*post** text box

*if **non blank**, pre\*post text to use for the models in this 12d XML file (see [4.19.2 Pre\\*Postfix in Panel Fields](#)) for information on using pre\*postfix.*

*If **blank**, use the pre\*post text from the Pre\*postfix for models panel field.*

**Note** - if a non-blank value for **Pre\*post** is given in the column for a file then the **Pre\*postfix for models** is ignored.

**File to read** file box \*.12dxmzlz, \*.12dxml

*name of the **12d Model** XML file to be read in.*

*Note that **Drag and Drop** works for 12dxmzlz and 12dxml files.*

*It will also work for files that are email attachments in Outlook 2002 and above.*

*If you are dragging and dropping more than one **12dxmzlz** or **12dxml** file at a time, then one panel will be opened and all the dropped files listed in the **Many Files** grid.*

**Map File** file box \*.mapfile, \*.mf files

*if **not blank**, the name of the 12d Map File to be used for all strings read in, including any files given with the **Many files** mode ticked on.*

*If **blank**, no map file is used*

*When using a map file, the string name is used as the entity-name for matching with the keys in the map file.*

**Pre\*postfix for models** pre\*postfix box

*if non-blank, a prefix and a postfix to be applied to the model names used in the map file.*

*Go to the section [4.19.2 Pre\\*Postfix in Panel Fields](#) for information on using pre\*postfix.*

**Note** - if a non-blank value for **Pre\*post** is given in the column for a file then the **Pre\*postfix for models** is ignored.

**Use pre\*postfix for tins** tick box not ticked

*if ticked, a prefix and a postfix are to be applied to any tin names in the 12d XML data.*

*Go to the section [4.19.2 Pre\\*Postfix in Panel Fields](#) for information on using pre\*postfix.*

**Warning** - if a tin already exists in 12d Model with the tin name, then the tin cannot be read in from the 12d XML file.

**Use map file model when pt/line changes** tick box

*if **not ticked** and the pt/line type of the string does not match that in the map file, then the string is placed in.*

*If **tick**, the.*

**Allow #include to be used** tick box

*if ticked, expand an files referenced by an #include.*

*If not ticked, ignore the #includes*

**Convert 2d, 3d, 4d, poly, face, interface to super** tick box

*if ticked, non super string versions of 2d/3d/4d/poly/face/interface strings are converted to super strings.*

**Read** button

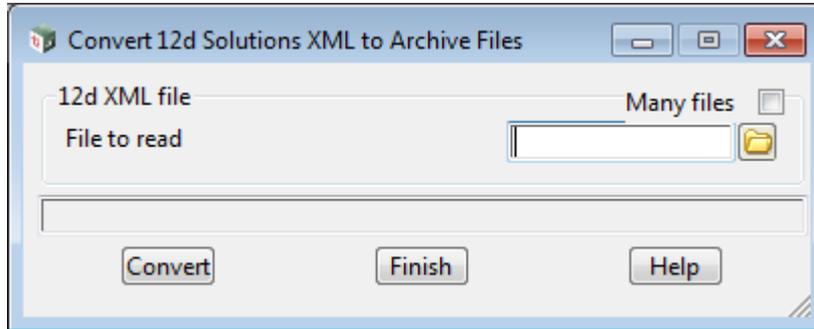
*reads the data in.*

### 15.1.3 12d XML to 12da files

**Position of option on menu:** File I/O =>Data input =>12d =>12d XML to 12da files

The **12d XML to 12da** option converts **12d XML** files to **12da** files.

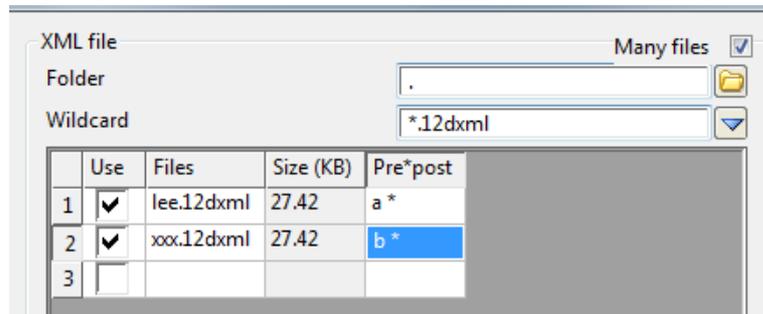
Selecting 12d XML to 12da files brings up the **Convert 12d Solutions XML to Archive Files** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Many files</b>	tick box		

*if **ticked**, a grid to allow multiple 12d XML files to be converted to 12d XML files. A wild card is used to select all the files to be converted.*



<b>Folder</b>	folder box
---------------	------------

*folder to search for files using the Wild card*

<b>Wildcard</b>	text box
-----------------	----------

*wild card to use in search for files in the given folder*

<b>Use</b>	tick box
------------	----------

*if **ticked**, convert the file*

<b>Files</b>	file box
--------------	----------

*name of the file in the folder*

<b>Size (KB)</b>	output only
------------------	-------------

*file size*

<b>Pre*post</b>	text box
-----------------	----------

*if **non blank**, pre\*post text.*

*If **blank**, use.*

<b>Convert</b>	button
----------------	--------

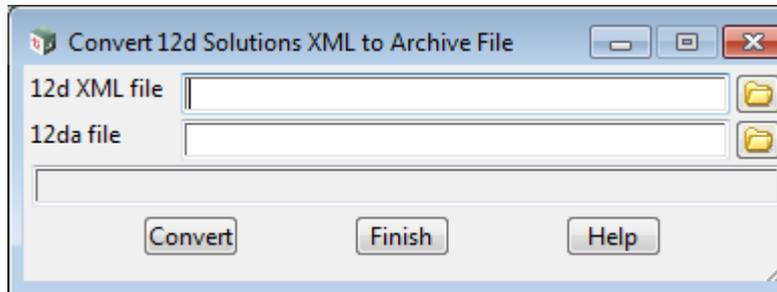
*converts the 12d XML files to 12da files with the same name stem.*

## 15.1.4 12d XML to 12da file

**Position of option on menu:** File I/O =>Data input =>12d =>12d XML to 12da file

The **12d XML to 12da** option converts **12d XML** file to a **12da** file.

Selecting 12d XML to 12da file brings up the **Convert 12d Solutions XML to Archive File** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>XML file</b> <i>name of the 12d XML file to convert to a 12da file.</i>	file box		*.12dxmlz, *.12dxml
<b>12da file</b> <i>name of the 12da file to convert the 12d XML file to.</i>	file box		*.12a files
<b>Convert</b> <i>convert the 12d XML file to the 12da file.</i>	button		

## 15.1.5 DWG PolyFace Mesh to Trimesh

Any PolyFace Meshes in the DWG file will be read in as Trimeshes.

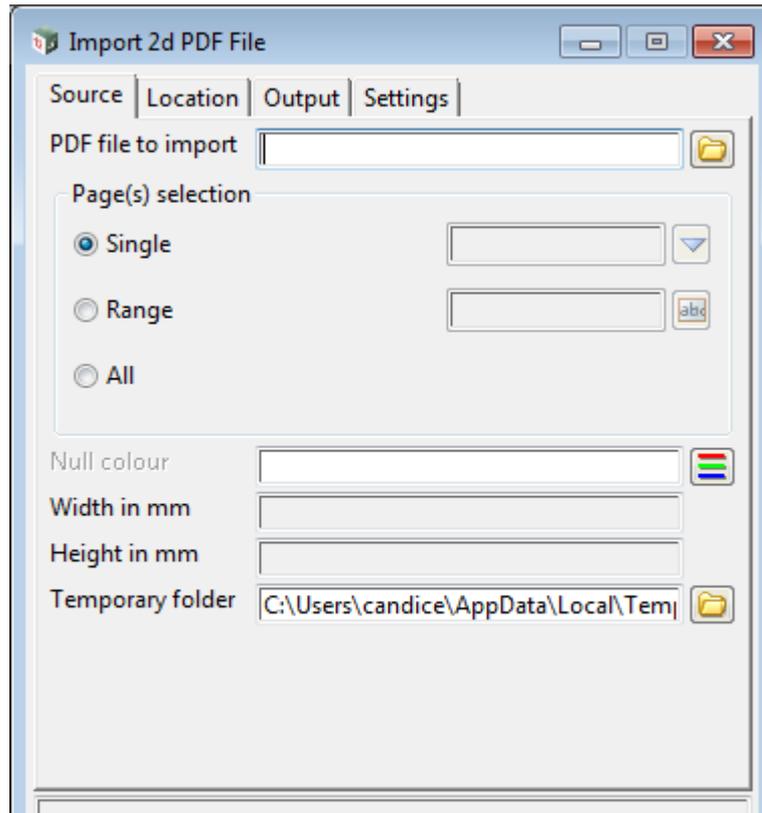
## 15.1.6 Import 2D PDF

This is a new option that reads in a 2d pdf file and can create an image of the pdf and/or extract any line work (vectors) and text that may exist in the 2d pdf. The option is currently at

**File =>Data input =>Import 2d pdf**

**Note:** Although you can "see" lines and text in the raster does not mean that they exist inside the pdf as lines (vectors) and text data. They may be in the image only. This option **does not** do text and/or line recognition from the raster.

Selecting **Import 2d pdf** displays the **Import 2d PDF File** panel.



The fields and buttons used in this panel have the following functions:

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

### Source tab

<b>PDF file to import</b>	file		*.pdf files
---------------------------	------	--	-------------

*name of the PDF file to import/read in.*

### Page(s) selection

*only one of **Single**, **Range** or **All** can be selected.*

<b>Single</b>	radio button and choice box	page numbers
---------------	-----------------------------	--------------

*if selected, this is the page number to import from the pdf file. The pop up has a list of pages and the page size (in mm) of each page in the pdf.*

<b>Range</b>	radio button, number box
--------------	--------------------------

*if selected, the range of the page numbers to read in from the pdf file.*

*The page numbers and/or page ranges are separated by commas, and the pages in a range are separated by a minus. For example 1,3,5-12*

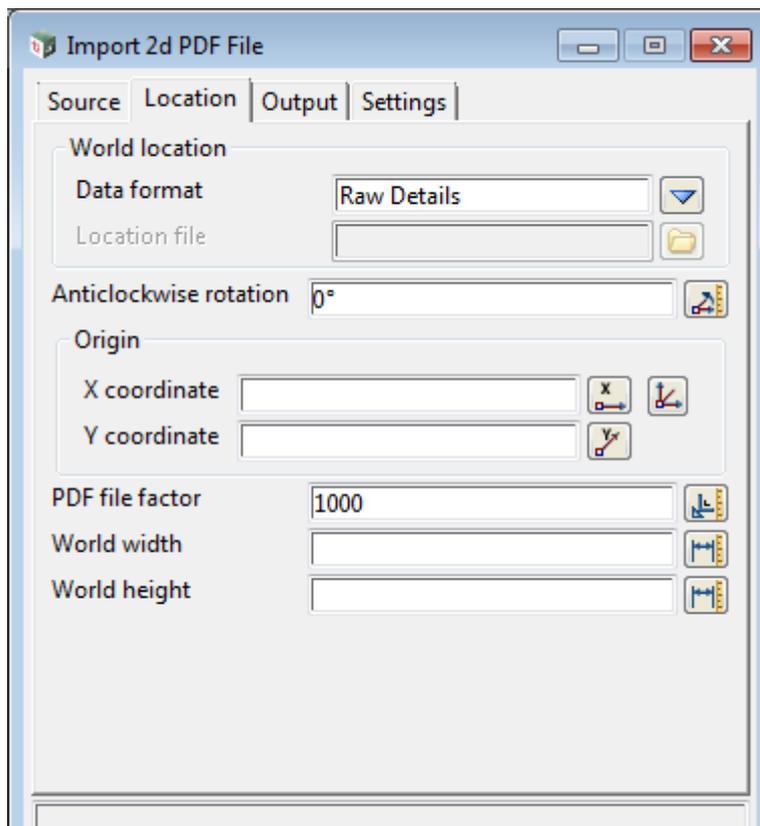
**All** radio button  
*if selected, all the pages in the pdf file are imported.*

**Null colour** colour box  
*if not blank, the colour to use for the background colour in the pdf images.  
 If blank, leave the colours as they are in the pdf.*

**Width/Height in mm** output only  
*the width/height (in mm) of the page selected from the pdf is written out to these fields.*

**Temporary folder** folder box  
*the folder to use for temporary files created whilst processing the pdf.*

**Location tab**



*The units in the pdf file are paper units and not world coordinates. The location tab supplies the information for positioning the data in the pdf file in world units by either typing in a world origin, anticlockwise rotation angle and world width and height, or by using supplying a file in **ESRI world** format with the information in it.*

**Data format** choice box      Raw details      Raw Details or ESRI world  
*if **Raw details**, the location details are typed into the **World Location Details** section as rotation, world origin, world width and height.*

*If **ESRI world**, the location details are taken from the **ESRI world** file given in **Location file**. An **ESRI world** file gives the *x*scale, row rotation, column rotation, *y*scale, *x* origin and *y* origin. For use with 12d, the row and column rotations must be the same and *y*scale = - *x*scale.*

**Location file** file box  
*file with the location details if **ESRI world** is selected as the **Data format**.*



vectors and the text placement positions.

**Vectors**  tick box

*if ticked, any vectors (line work) in the pdf file are read in and placed in the model given by the field **Name stem**.*

**Boundaries**  tick box

*if ticked, any polygon boundaries in the pdf file are read in and placed in the model given by the field **Name stem** followed by "**ClipBnd**".*

**Text**  tick box

*if ticked, any text in the pdf file is read in and placed in the model given by the field **Name stem** followed by "**Text**".*

**Create Raster**

*only one of **Colour**, **Greyscale** or **Mono** can be ticked. If none are ticked then no rasters are created from the pdf.*

*Any created rasters are placed in a model given by the field **Name stem** followed by " **raster**".*

**Colour**  tick box

*if ticked, a raster for each selected page of the pdf is created and the colours in the pdf are retained in the raster.*

**Greyscale**  tick box

*if ticked, a raster for each selected page of the pdf is created and the colours in the rasters are grey scaled.*

**Mono**  tick box

*if ticked, a raster for each selected page of the pdf is created and any colours are mapped to black.*

**Name stem**  text box name of the pdf file with each non alphanumeric character replaced by a space

*the name stem used to generate the model names for the created models.*

**Include page number in names**  tick box  ticked

*if ticked, the page numbers in the pdf are also included as part of the model names.*

**Clean models first**  tick box  ticked

*if ticked, the models are cleaned before the pdf pages are read in.*

**View for output**  view box

*if non-blank, add the models to this view*

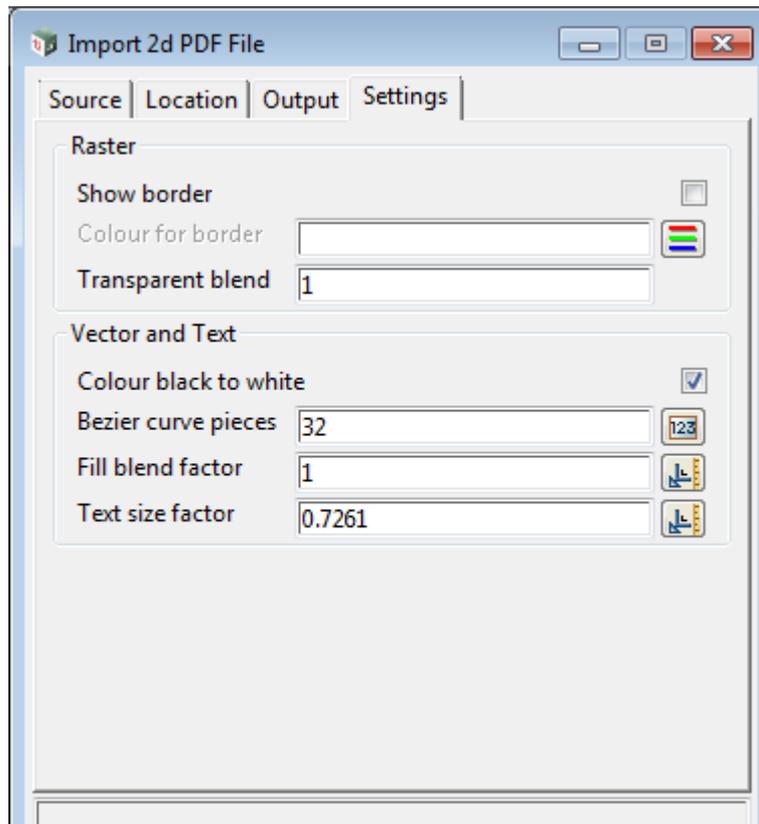
**Output Raster DPI**  number box 150

*the number of dots per inch for the created rasters.*

**Raster size**  output

*the size of the produced raster.*

## Settings tab



### Raster

*Settings to use for rasters.*

**Show border**                      tick box  
*if ticked, a border is drawn around the raster.*

**Colour for border**              colour box                              available colours  
*colour to draw the border around the raster:*

**Transparent blend**              real box                                      1  
*blend (transparency) to apply to the rasters. The value is between 0 and 1 and 1 means it is fully opaque (i.e. no transparency).*

### Vector and Text

*Settings to use for vectors (line work) and text.*

**Colour black to white**          tick box  
*if ticked, black colour are changed to white. Useful when using a black background for a view.*

**Bezier curve pieces**            number box                                      32  
*if there are Bezier curves in the vectors, then they are broken into this many linear segments.*

**Fill blend factor**                real box                                      1  
*extra Blend (transparency) to apply to polygon fills. The value is between 0 and 1 and 1 means it is fully opaque (i.e. no transparency).*

**Text size factor**                real box                                      0.7261  
*the text size in pdf is different to traditional 12d text sizing (pdf size includes descenders). This factor is applied to any text to correct for the difference in size definitions.*

## **Button at bottom**

**Create**

button

*import the 2d PDF file into the appropriate model.*

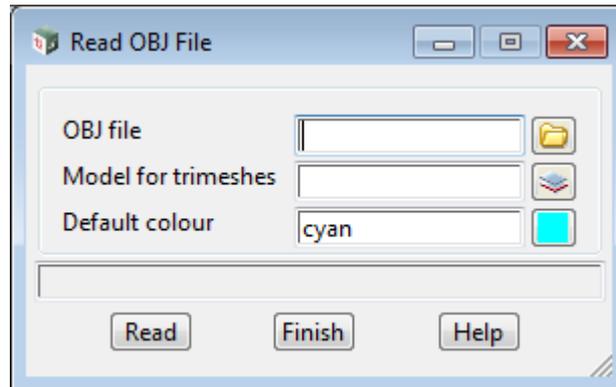
## 15.1.7 Wavefront OBJ Input

**Position of option on menu:** File I/O =>Data input =>OBJ

**Position of option on menu:** Strings =>Trimesh =>Read OBJ file

The **OBJ** option reads in a Wavefront OBJ file and creates trimeshes.

Selecting **OBJ** brings up the **Read OBJ File** panel.



The fields and buttons used in this panel have the following functions.

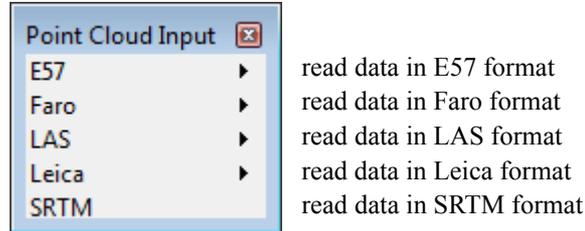
Field Description	Type	Defaults	Pop-Up
<b>Obj file</b> <i>name of the OBJ file to read in</i>	file box		*.OBJ files
<b>Model for trimeshes</b> <i>model to put the created trimeshes in.</i>	model box		available models
<b>Default colour</b>	colour box		available colours
<b>Read</b> <i>read the obj data in</i>	button		

## 15.1.8 Input Point Cloud Files

**Position of menu:** File I/O =>Data input =>Point clouds

The options under **Point Cloud Input** read data in various formats.

The **Point Cloud Input** walk-right menu is



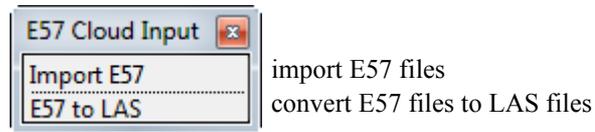
See

- E57*      [15.1.8.1 E57 Cloud Input](#)
- Faro*     [15.1.8.2 Faro Point Cloud Input](#)
- LAS*     [15.1.8.3 LAS Cloud Input](#)
- Leica*    [15.1.8.4 Leica Point Cloud Input](#)
- SRTM*    [15.1.8.5 Read SRTM Files](#)

## 15.1.8.1 E57 Cloud Input

**Position of menu:** File I/O =>Data input =>Point clouds =>E57

The E57 Cloud Input walk-right menu is



See

*Import E57*

[15.1.8.1.1 Import E57 Files](#)

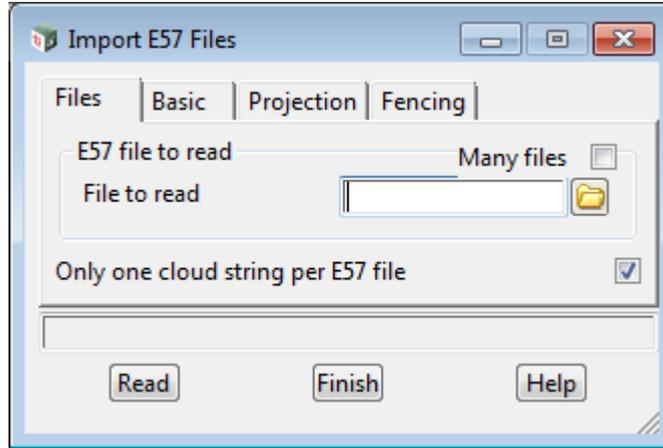
*E57 to LAS*

[15.1.8.1.2 E57 to LAS Files](#)

### 15.1.8.1.1 Import E57 Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>E57 =>Import E57

Selecting **Import E57** brings up the **Import E57 Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

**Many files**                      tick box                      not ticked

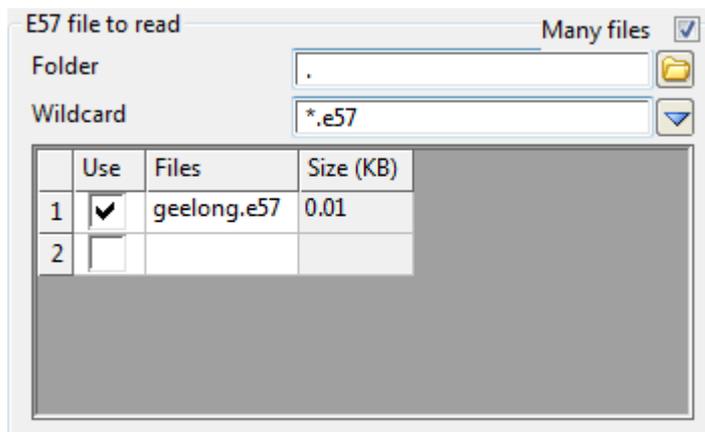
*if **ticked**, a grid to allow multiple e57 files to be read in is opened. A wild card is used to select all the files to be read in.*

**Folder**                              file box

*folder to search for files using the Wild card*

**Wildcard**                              input

*wild card to use in search for files in the given folder*



**Use**                                      tick box

*if **ticked**, read in the file*

**Files**                                      output

*name of the file in the folder*

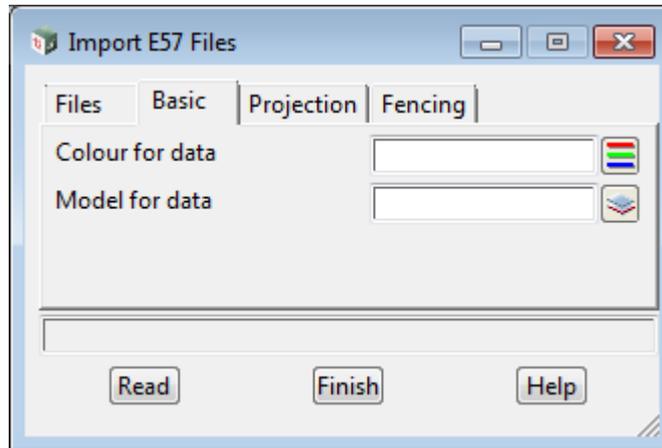
**Size** output

*file size*

**Only one cloud string per E57 file** tick box  ticked

*if ticked, all cloud points will be combined into one 12d cloud string.*

**Basic tab**



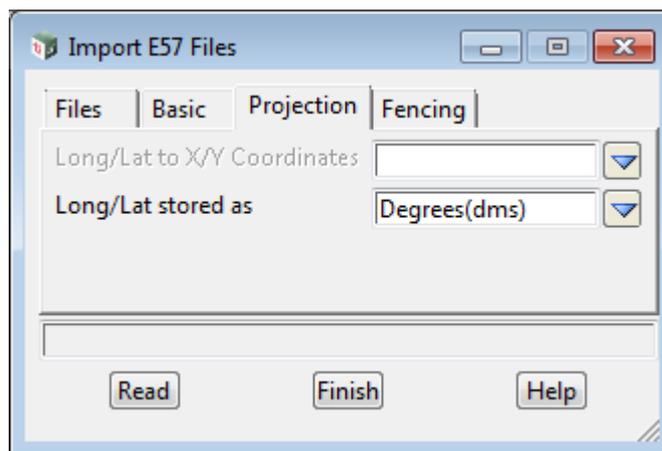
**Colour for data** colour box

*string colour to be used if the cloud object does not have colour for each point*

**Model for data** model box

*name of the model that the data is to be placed in. The model will be created if it does not already exist. This field must be filled in.*

**Projection tab**



**Long/Lat to X/Y Coordinates** choice box available projections

*if non-blank, the cartographic projection to apply to the longitude-latitude values.*

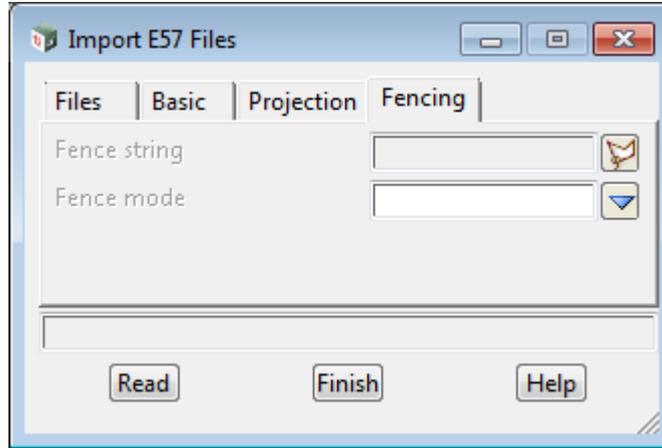
*If blank, the co-ordinates are not transformed from (longitude, latitude) to (x,y). Hence the initial (x,y) co-ordinates are transformed to (longitude, latitude) by the transformation given in the **xly co-ordinates to Long/Lat** field and then left in (longitude, latitude). Note that in the southern hemisphere, the latitude values are **negative**.*

**Long/Lat stored as** choice box degrees radians degrees

decimal degrees

*format for the longitude and latitudes - either radians, degrees (in [4.17.1 HP Notation](#) for degrees, minutes and seconds) or decimal degrees.*

**Fencing tab**



**Fence string**                      polygon box

*string to use to restrict the data being read in.*

**Fence mode**                      choice box

- File inside
- File inside/crossing
- File outside
- File outside/crossing
- File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

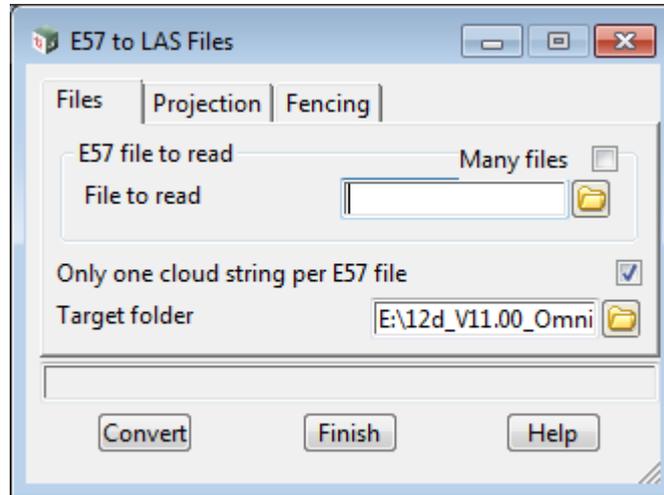
**Read**                                  button

*read the data into the model given in the model field.*

### 15.1.8.1.2 E57 to LAS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>E57 =>E57 to LAS

Selecting E57 to LAS brings up the **E57 to LAS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

**Many files**                      tick box                      not ticked

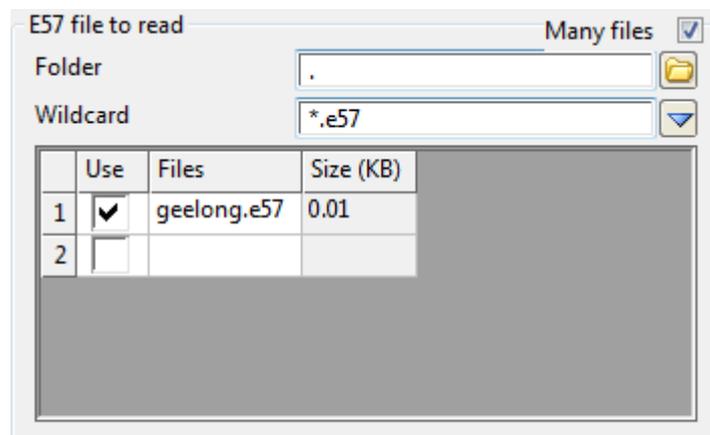
*if **ticked**, a grid to allow multiple e57 files to be converted is opened. A wild card is used to select all the files to be read in.*

**Folder**                              file box

*folder to search for files using the Wild card*

**Wildcard**                              input

*wild card to use in search for files in the given folder*



**Only one cloud string per E57 file** tick box    ticked

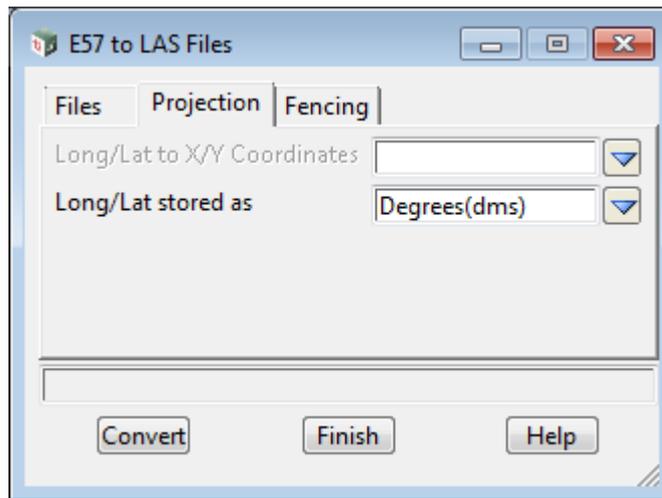
*if ticked, all cloud points will be combined into one 12d cloud string.*

**Target folder** file box

folder where the output LAS file(s) will be created. As it is unknown how many files will be created, it is good practice to specify an empty folder for these files. For example, if the input file is 'geelong.e57', the output file(s) will be 'geelong.las' OR 'geelong\_1.las', 'geelong\_2.las', etc.

Note: Multiple las files will only be created if the tick box 'Only one cloud string per file' is unticked and the input file has multiple clouds.

**Projection tab**



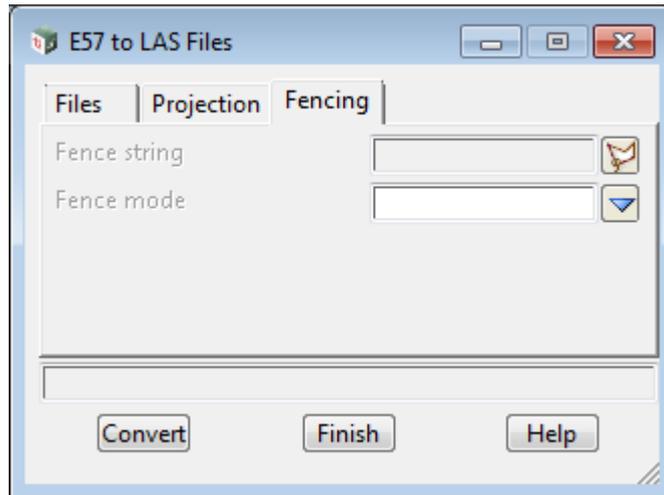
**Long/Lat to X/Y Coordinates** choice box available projections

if **non-blank**, the cartographic projection to apply to the longitude-latitude values.  
 If **blank**, the co-ordinates are not transformed from (longitude, latitude) to (x,y). Hence the initial (x,y) co-ordinates are transformed to (longitude, latitude) by the transformation given in the **x/y co-ordinates to Long/Lat** field and then left in (longitude, latitude). Note that in the southern hemisphere, the latitude values are **negative**.

**Long/Lat stored as** choice box degrees radians degrees decimal degrees

format for the longitude and latitudes - either radians, degrees (in [4.17.1 HP Notation](#) for degrees, minutes and seconds) or decimal degrees.

**Fencing tab**



**Fence string**                      polygon box  
*string to use to restrict the data being read in.*

**Fence mode**                      choice box

- File inside
- File inside/crossing
- File outside
- File outside/crossing
- File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

***Note** - only whole files are read in.*

**Button at bottom**

**Convert**                      button  
*convert the data into the model given in the model field.*

## 15.1.8.2 Faro Point Cloud Input

**Position of menu:** File I/O =>Data input =>Point clouds =>Faro

This section of documentation is a work in progress and will be updated in subsequent releases.

The **Faro Point Cloud Input** walk-right menu is



See

*Import Faro FLS*

[15.1.8.2.1 Import Faro FLS Files](#)

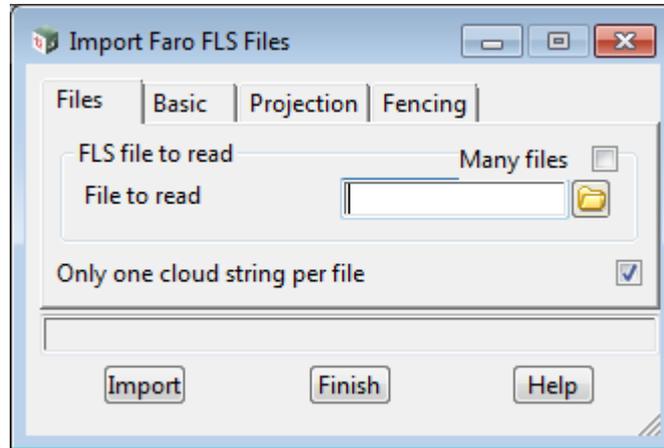
*Faro FLS to LAS*

[15.1.8.2.2 Faro Cloud FLS to LAS Files](#)

### 15.1.8.2.1 Import Faro FLS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>Faro =>Import Faro FLS

Selecting **Import Faro FLS** brings up the **Import Faro FLS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

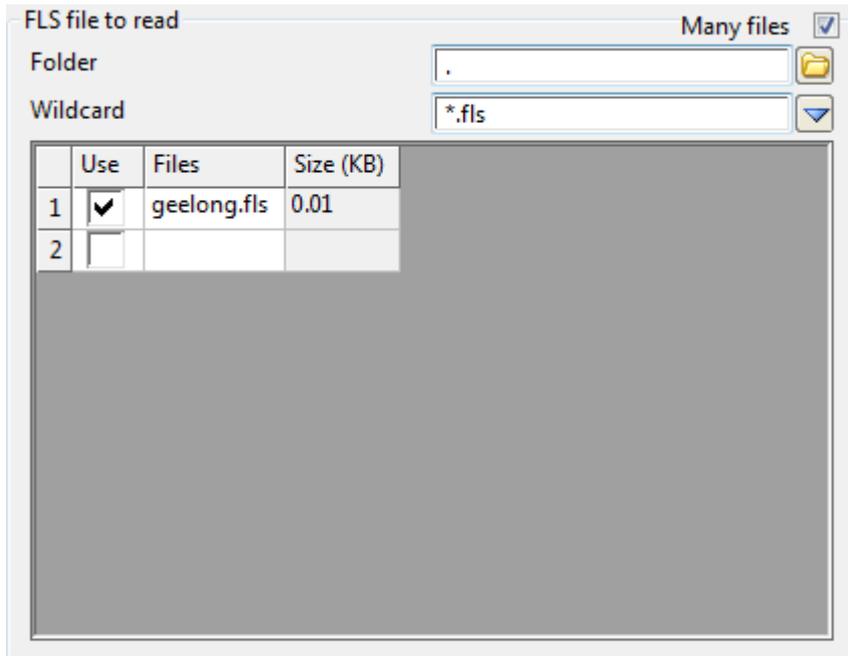
*if **ticked**, a grid to allow multiple fls files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



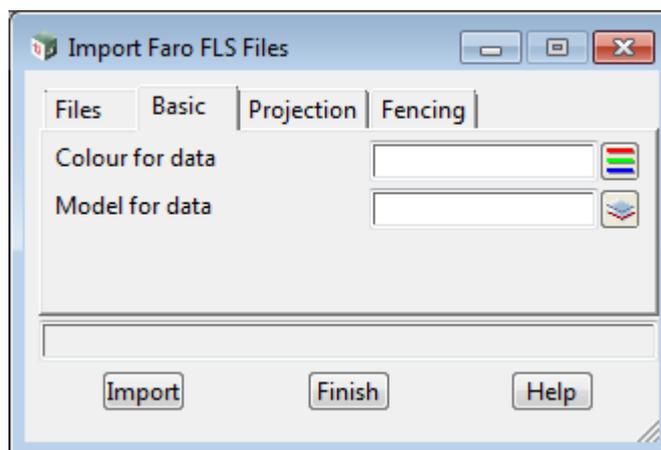
**Use** tick box  
 if *ticked*, read in the file

**Files** output  
 name of the file in the folder

**Size** output  
 file size

**Only one cloud string per file** tick box      ticked  
 if *ticked*, all cloud points will be combined into one 12d cloud string.

**Basic tab**



**Colour for data** colour box  
 string colour to be used if the cloud object does not have colour for each point

**Model for data** model box



File outside/crossing  
File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

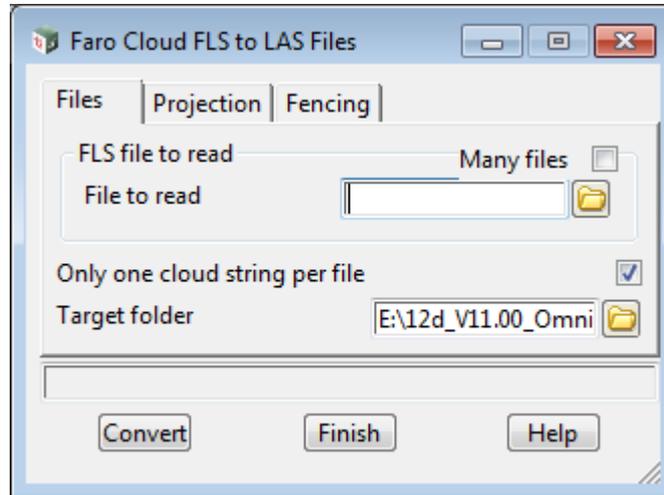
**Import** button

*import the data into the model given in the model field.*

### 15.1.8.2.2 Faro Cloud FLS to LAS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>Faro =>Faro FLS to LAS

Selecting Faro FLS to LAS brings up the **Faro Cloud FLS to LAS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

#### Files tab

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

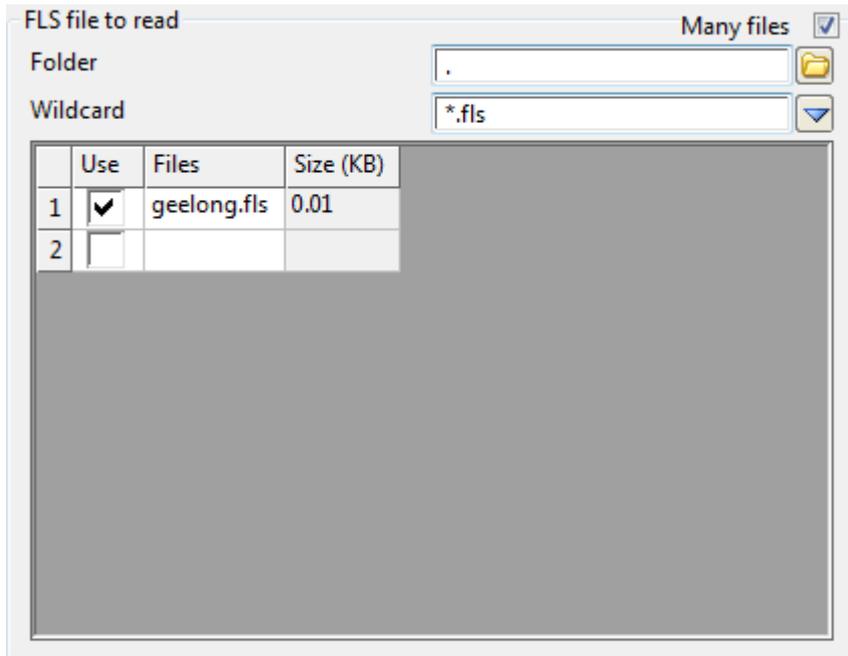
*if **ticked**, a grid to allow multiple fls files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if **ticked**, read in the file*

**Files** output

*name of the file in the folder*

**Size** output

*file size*

**Only one cloud string per file** tick box      **ticked**

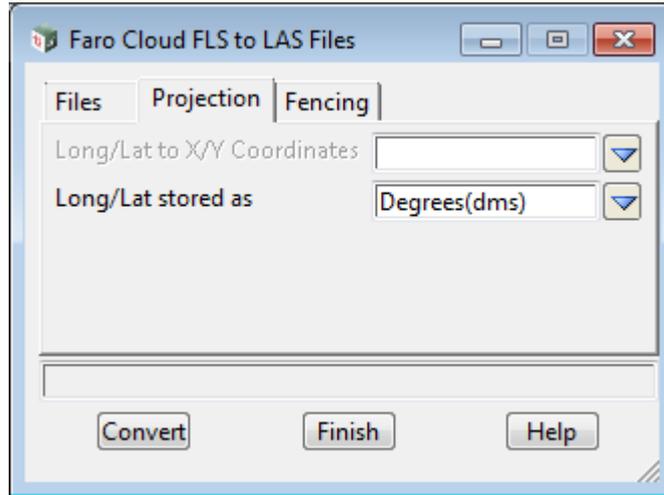
*if **ticked**, all cloud points will be combined into one 12d cloud string.*

**Target folder** file box

*folder where the output LAS file(s) will be created. As it is unknown how many files will be created, it is good practice to specify an empty folder for these files. For example, if the input file is 'geelong.flc', the output file(s) will be 'geelong.las' OR 'geelong\_1.las', 'geelong\_2.las', etc.*

*Note: Multiple las files will only be created if the tick box 'Only one cloud string per file' is **unticked** and the input file has multiple clouds.*

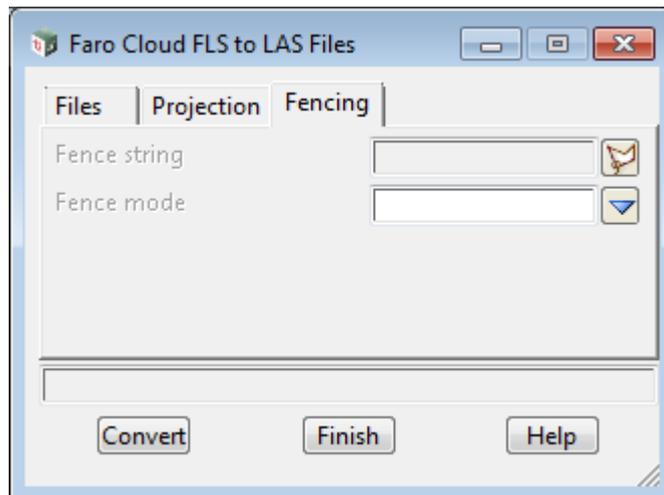
**Projection tab**



**Long/Lat to X/Y Coordinates** choice box available projections  
 if **non-blank**, the cartographic projection to apply to the longitude-latitude values.  
 If **blank**, the co-ordinates are not transformed from (longitude, latitude) to (x,y). Hence the initial (x,y) co-ordinates are transformed to (longitude, latitude) by the transformation given in the **x/y co-ordinates to Long/Lat** field and then left in (longitude, latitude). Note that in the southern hemisphere, the latitude values are **negative**.

**Long/Lat stored as** choice box degrees radians  
 degrees  
 decimal degrees  
 format for the longitude and latitudes - either radians, degrees (in [4.17.1 HP Notation](#) for degrees, minutes and seconds) or decimal degrees.

**Fencing tab**



**Fence string** polygon box  
 string to use to restrict the data being read in.

**Fence mode** choice box  
 File inside  
 File inside/crossing  
 File outside  
 File outside/crossing  
 File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

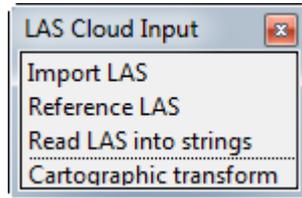
**Convert** button

*convert the data into the model given in the model field.*

### 15.1.8.3 LAS Cloud Input

**Position of menu:** File I/O =>Data input =>Point clouds =>LAS

The LAS Cloud Input walk-right menu is



import LAS files  
reference LAS files  
read LAS files into strings  
transform LAS Cartographic

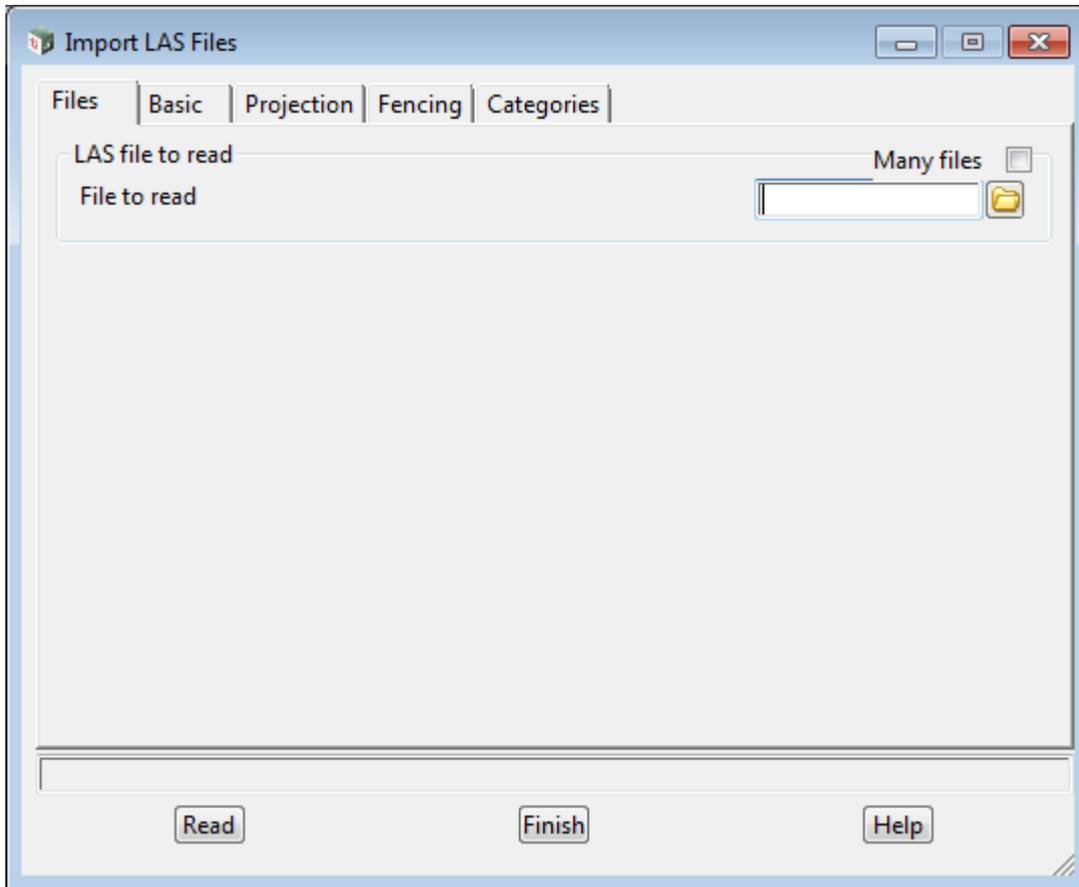
See

<i>Import LAS</i>	<a href="#">15.1.8.3.1 Import LAS Files</a>
<i>Reference LAS</i>	<a href="#">15.1.8.3.2 Reference LAS Files</a>
<i>Read LAS into strings</i>	<a href="#">15.1.8.3.3 Read LAS Files into Strings</a>
<i>Cartographic transform</i>	<a href="#">15.1.8.3.4 LAS Cartographic</a>

### 15.1.8.3.1 Import LAS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>LAS =>Import LAS

Selecting **Import LAS** brings up the **Import LAS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

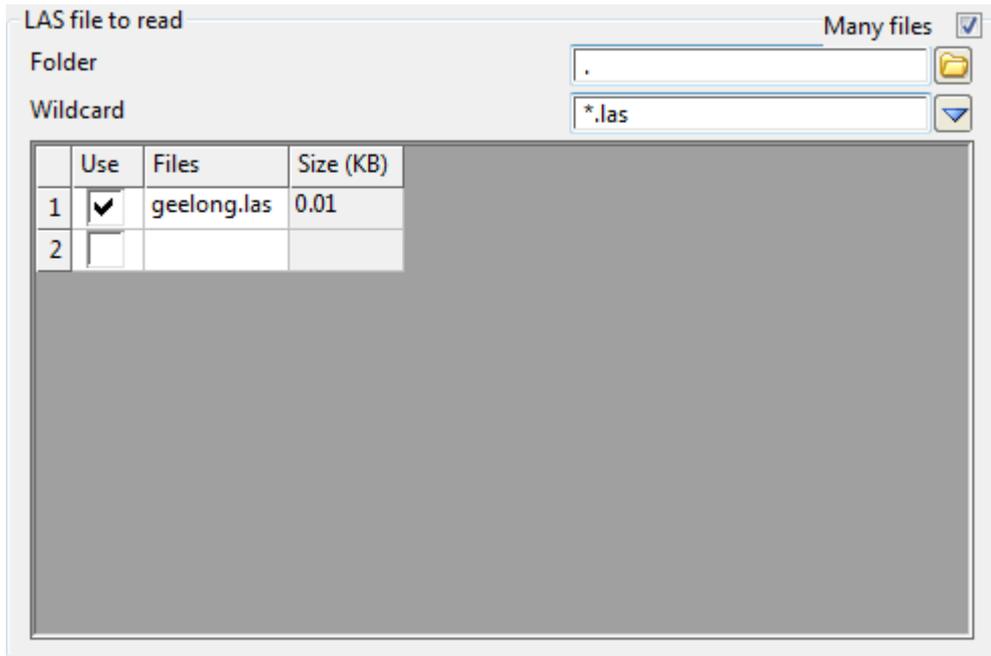
*if **ticked**, a grid to allow multiple las files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card.*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder.*



**Use** tick box

*if ticked, read in the file*

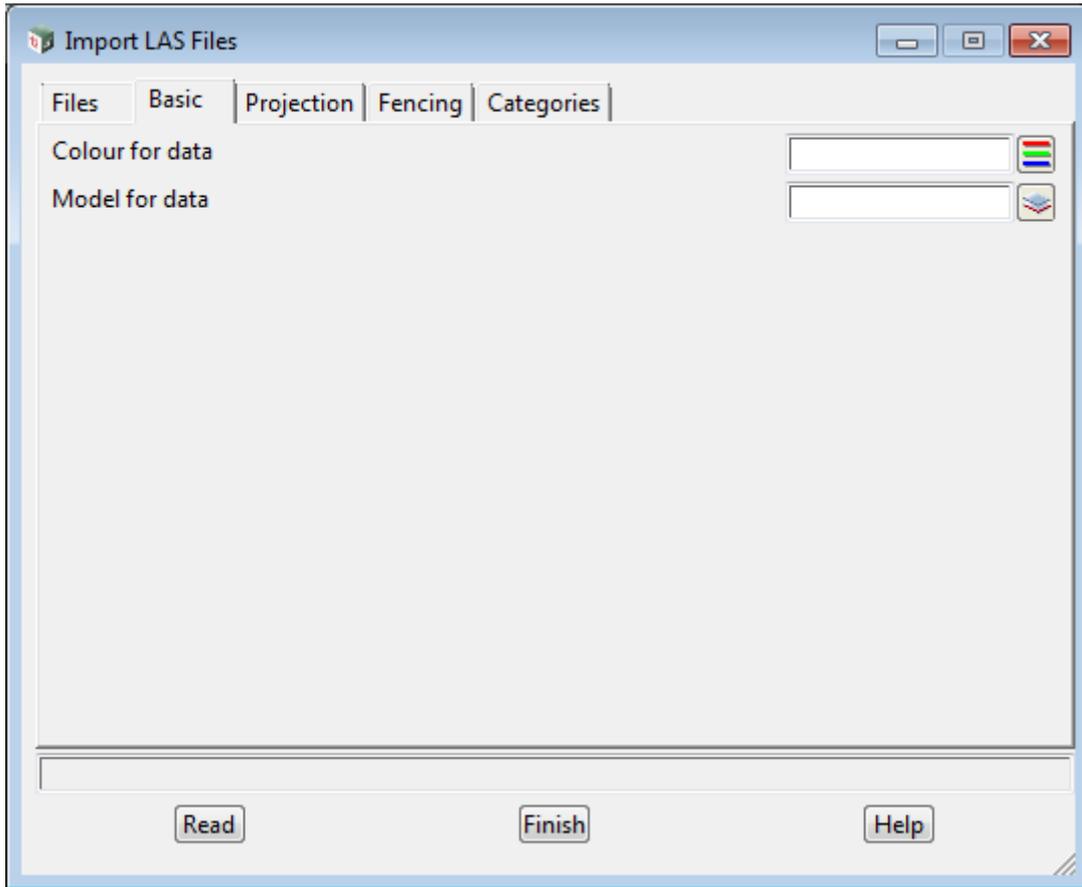
**Files** output

*name of the file in the folder*

**Size** output

*file size*

**Basic tab**



**Colour for data**                      colour box

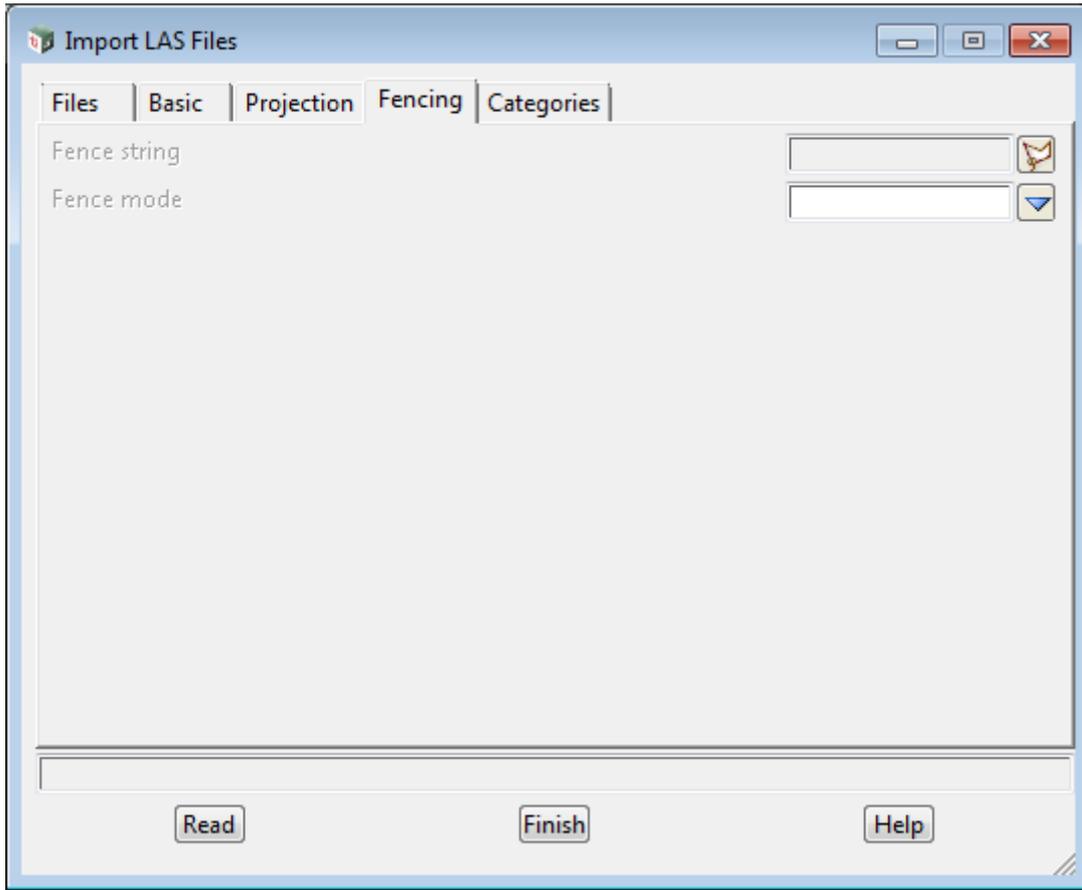
*string colour to be used if the cloud object does not have colour for each point*

**Model for data**                      model box

*name of the model that the data is to be placed in. The model will be created if it does not already exist. This field must be filled in.*

**Projection tab**





**Fence string**                      polygon box

*string to use to restrict the data being read in.*

**Fence mode**                      choice box

- File inside
- File inside/crossing
- File outside
- File outside/crossing
- File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

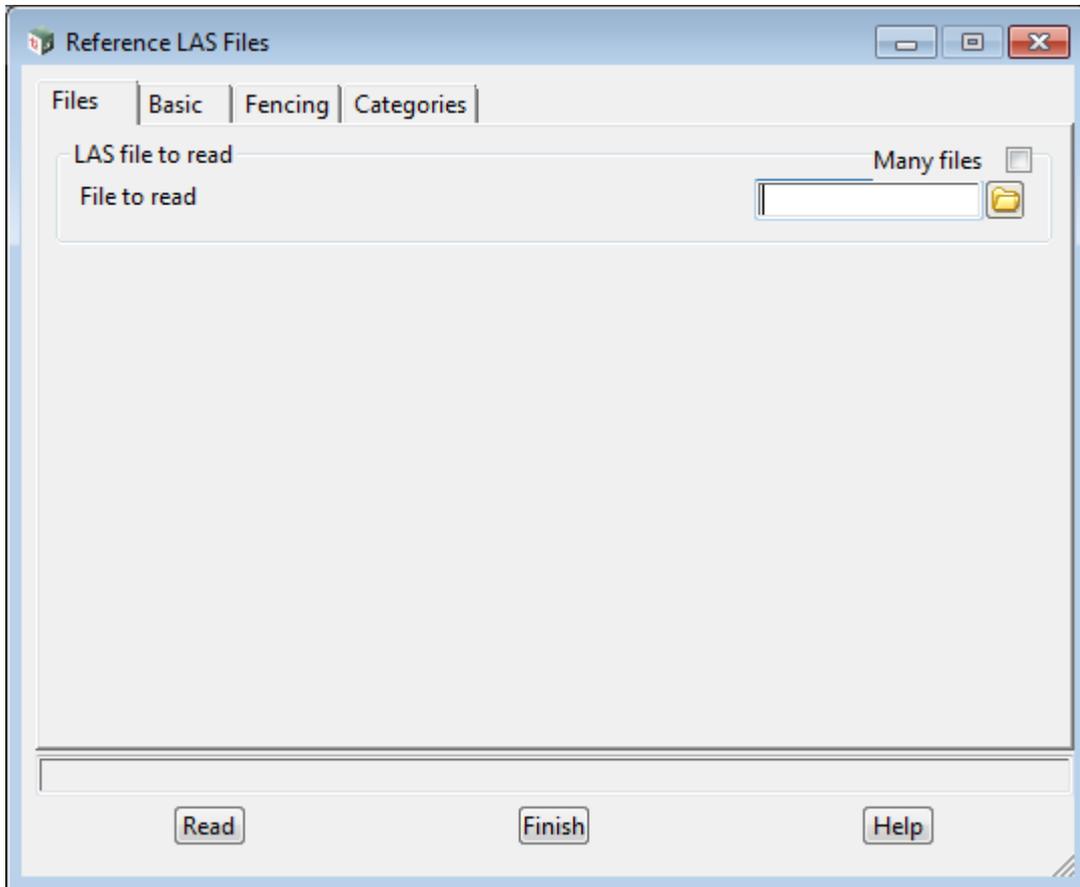
**Categories tab**



### 15.1.8.3.2 Reference LAS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>LAS =>Reference LAS

Selecting Reference LAS brings up the **Reference LAS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

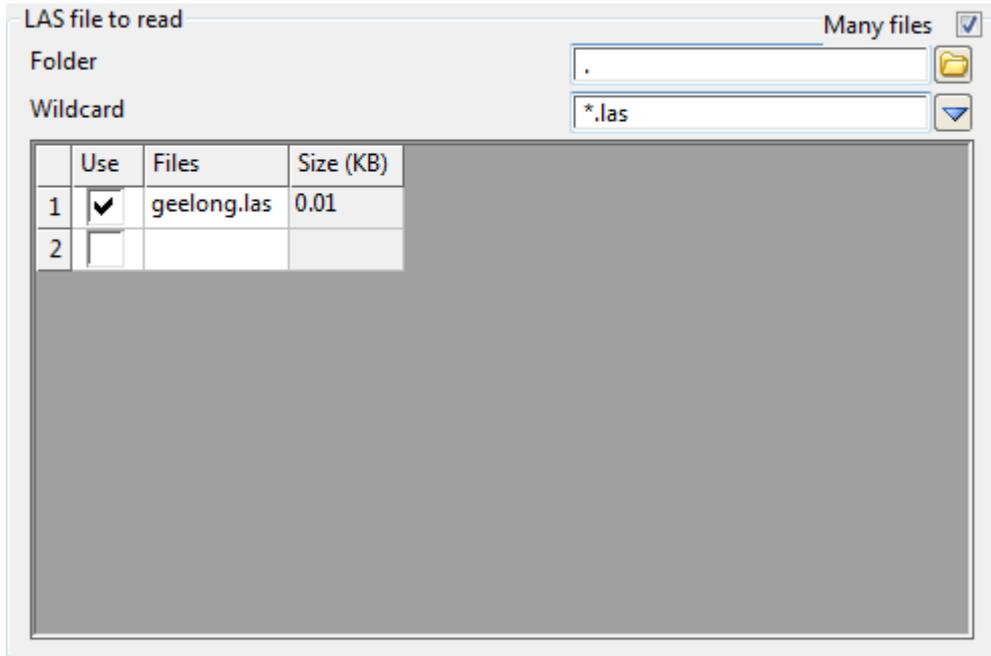
*if **ticked**, a grid to allow multiple las files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if ticked, read in the file*

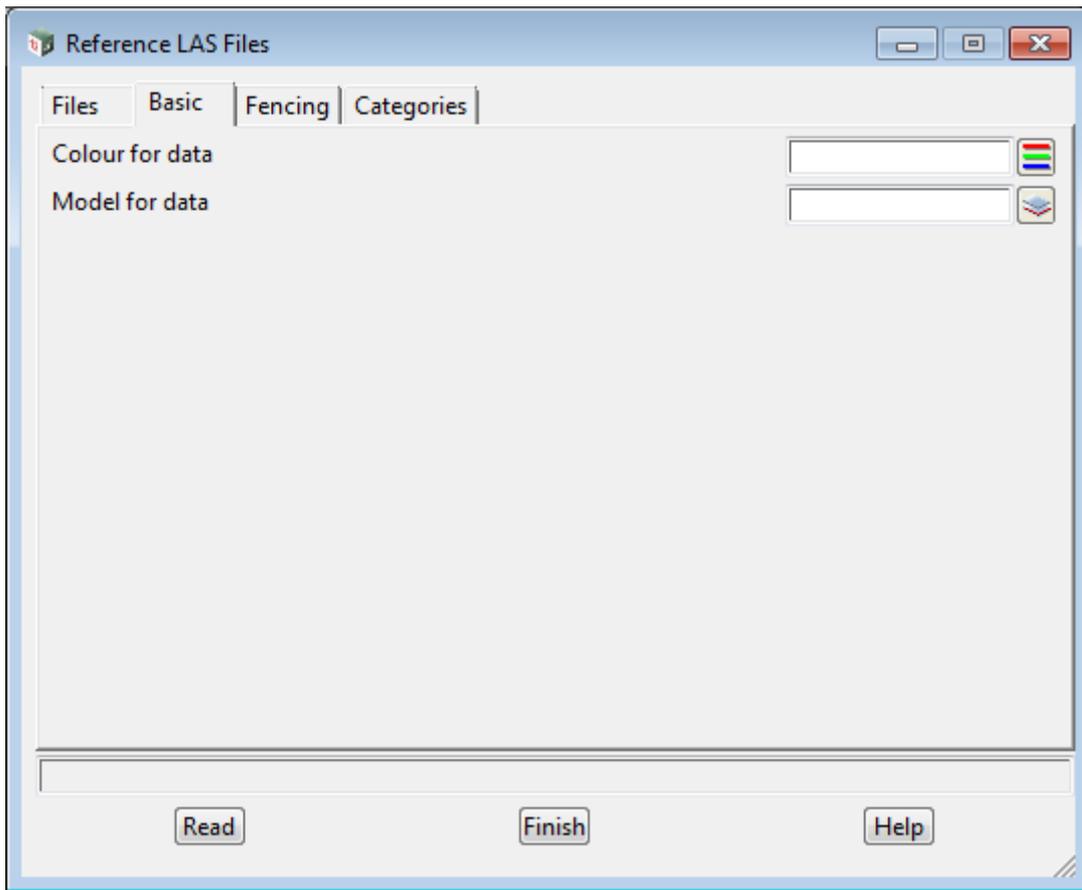
**Files** output

*name of the file in the folder*

**Size** output

*file size*

**Basic tab**



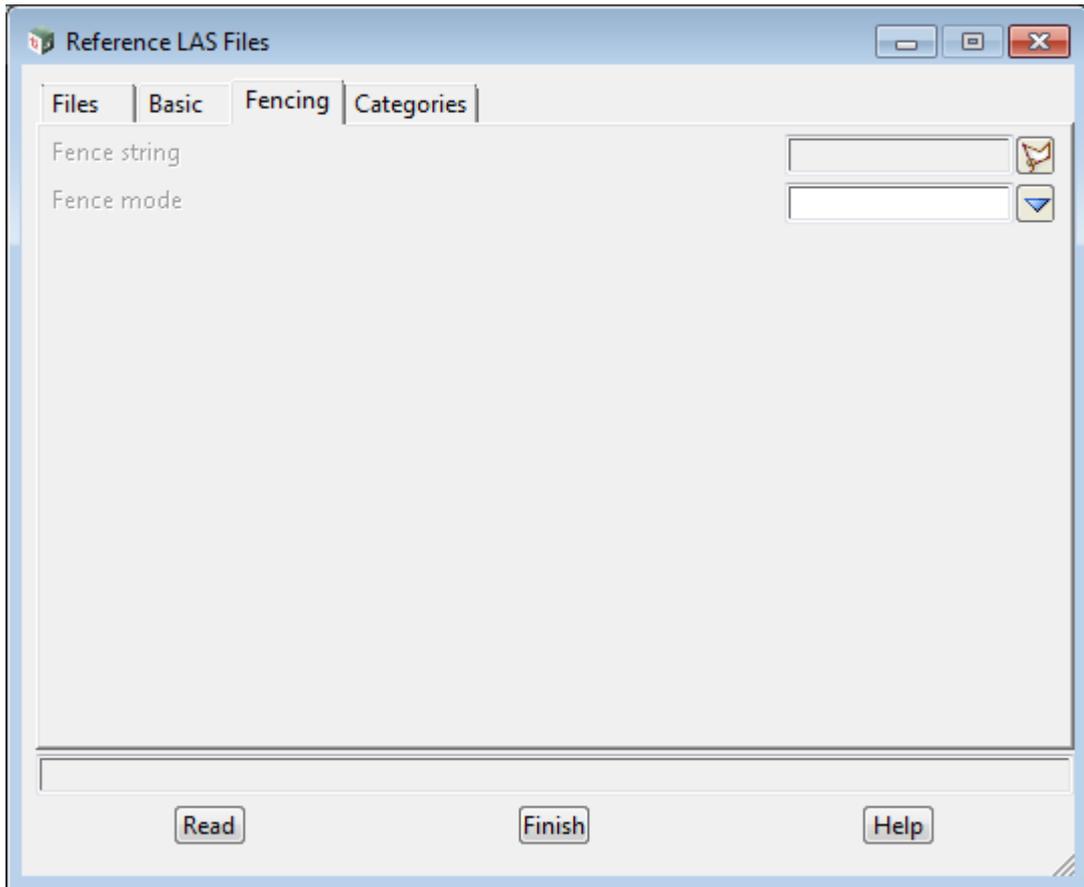
**Colour for data**                      colour box

*string colour to be used if the cloud object does not have colour for each point*

**Model for data**                      model box

*name of the model that the data is to be placed in. The model will be created if it does not already exist. This field must be filled in.*

**Fencing tab**



**Fence string**                      polygon box  
*string to use to restrict the data being read in.*

**Fence mode**                      choice box

- File inside
- File inside/crossing
- File outside
- File outside/crossing
- File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

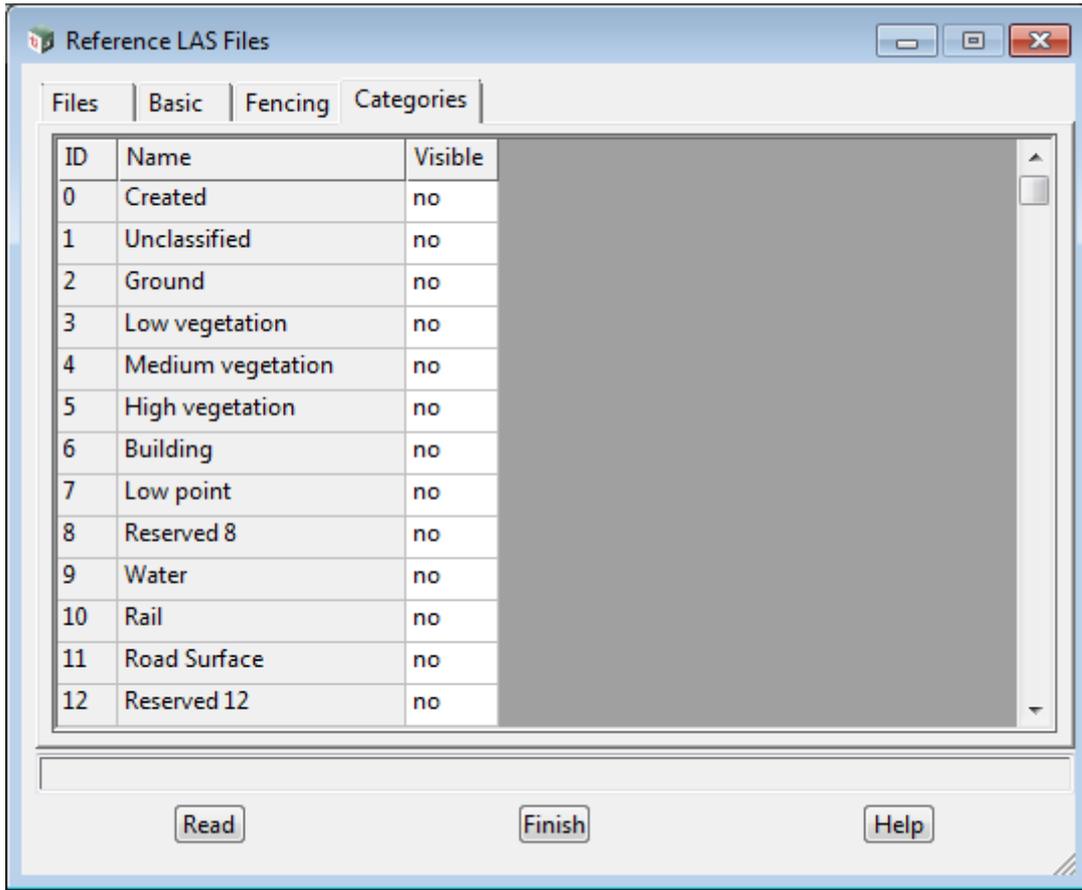
*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

**Note** - only whole files are read in.

**Categories tab**



**ID**

**Name**

**Visible**

**Button at bottom**

**Read**

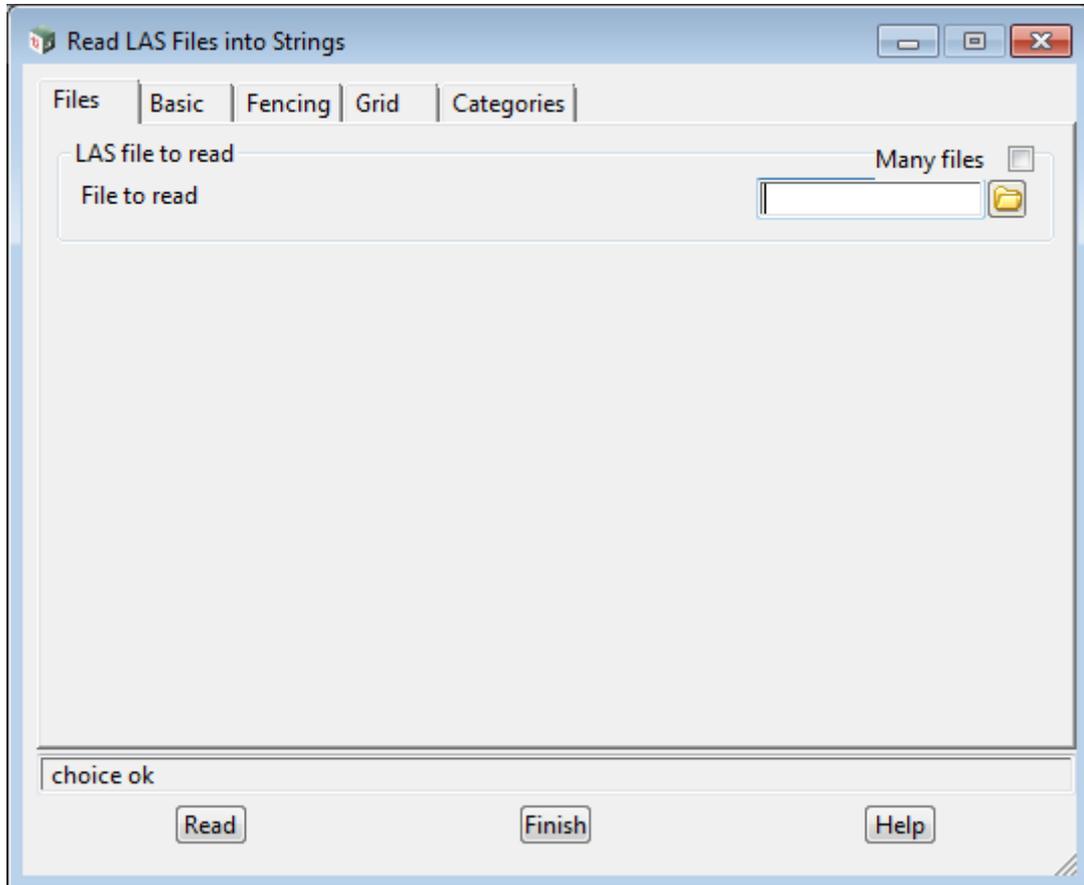
**button**

*read the data into the model given in the model field.*

### 15.1.8.3.3 Read LAS Files into Strings

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>LAS =>Read LAS into strings

Selecting Read LAS into strings brings up the **Read LAS Files into Strings** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

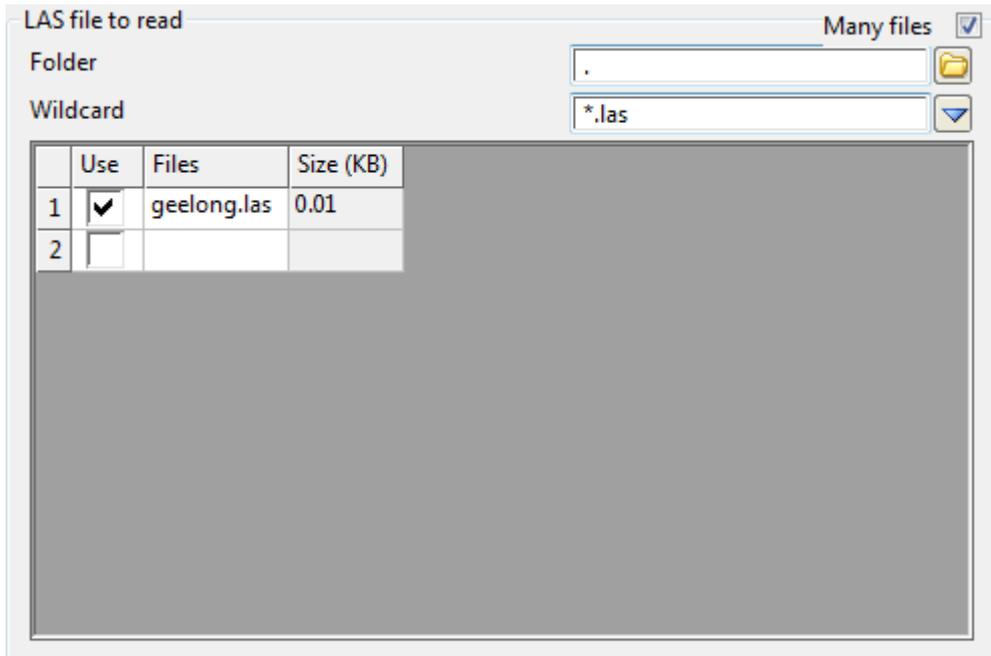
*if **ticked**, a grid to allow multiple las files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if ticked, read in the file*

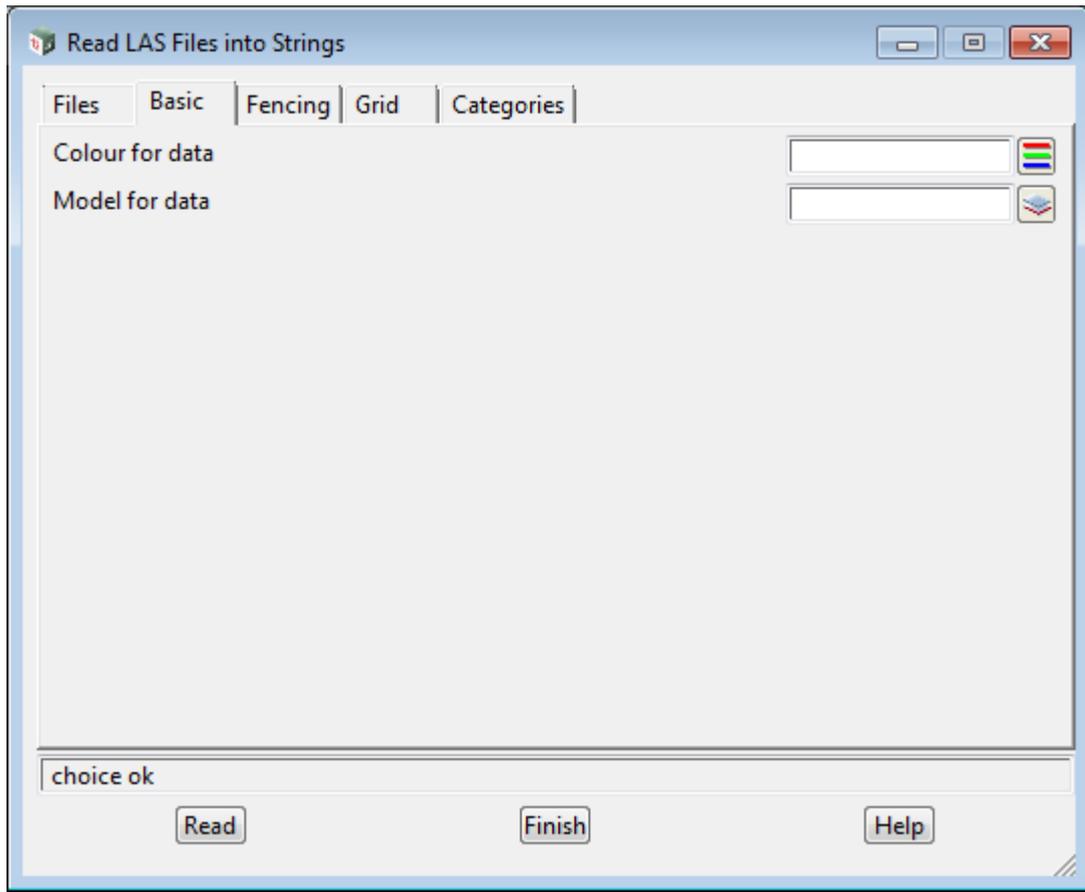
**Files** output

*name of the file in the folder*

**Size** output

*file size*

**Basic tab**



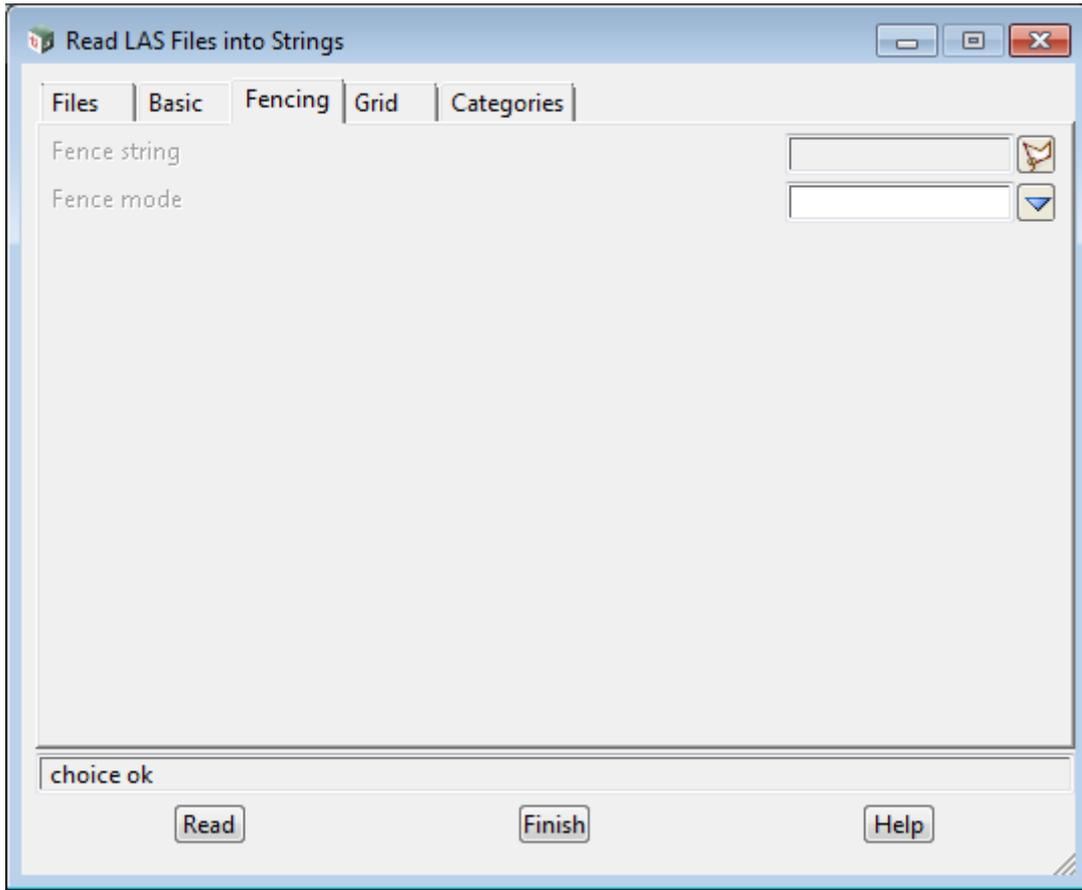
**Colour for data**                      colour box

*string colour to be used if the cloud object does not have colour for each point*

**Model for data**                      model box

*name of the model that the data is to be placed in. The model will be created if it does not already exist. This field must be filled in.*

**Fencing tab**



**Fence string**                      polygon box

*string to use to restrict the data being read in.*

**Fence mode**                      choice box

- File inside
- File inside/crossing
- File outside
- File outside/crossing
- File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

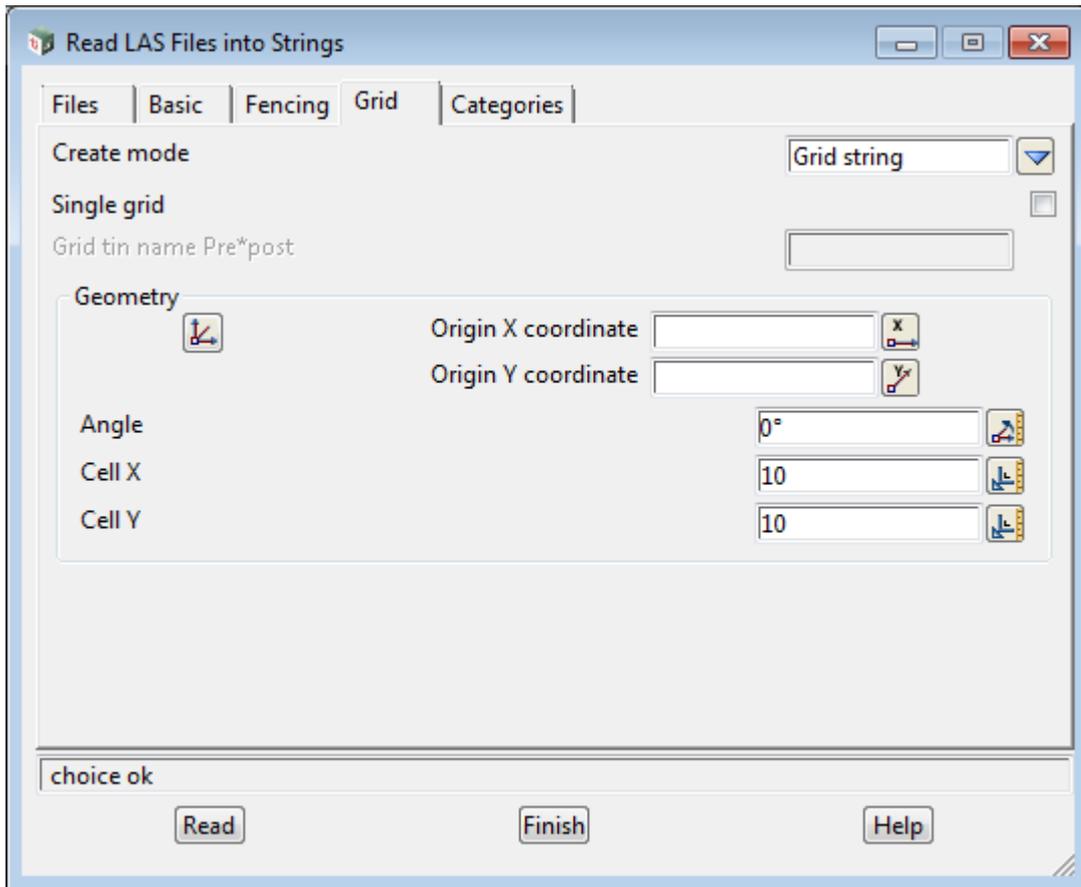
*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

**Note** - only whole files are read in.

**Grid tab**



**Create mode**                      choice box                      Grid string                      Grid string, Grid tin, Strings  
 see [15.10 Grids](#) for an explanation of Grid string and Grid tin

**Single grid**                      tick box                      not ticked  
 if ticked, all points from all selected files are used in the creation of a single grid.

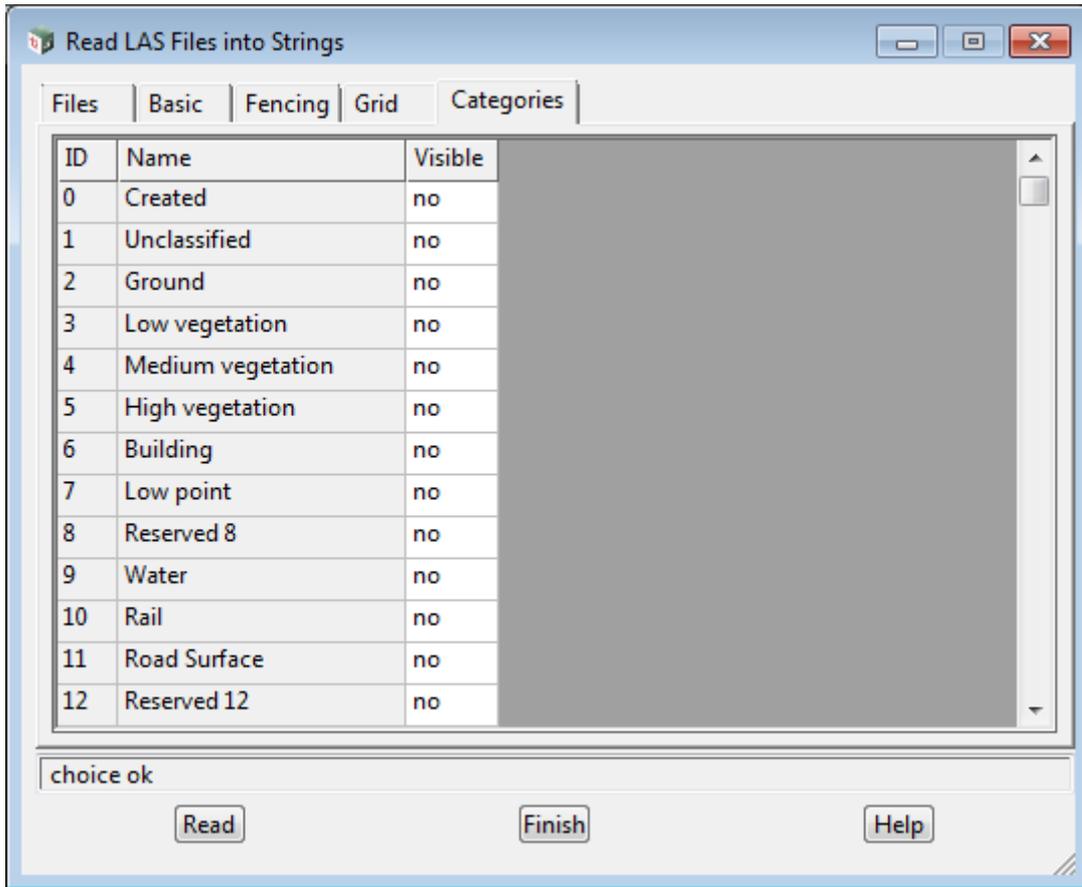
**Grid tin name Pre\*post**      input  
 only enabled if Grid tin is selected. All grid tins created will have their names derived from the text entered with the \* character beginning at 1 and incrementing for each tin created.

**Origin X/Y coordinate**      X/Y boxes  
 the Origin X and Y coordinates signify a seed point on the grid. The seed point does not have to be within the extents of the LAS points. The computed grid origin x/y will be within LAS points, and the origin x/y specified will lie on an infinite version of the resulting grid.

**Angle**                              measure box  
 the angle of the grid's local X axis

**Cell X/Y**                              measure boxes  
 the local X axis and Y axis spacings

### Categories tab



**ID**

*the unique number as defined by the LAS standard*

**Name**

*the description for the equivalent ID*

**Visible**

*if **yes**, the specific category will be active for all operations on that cloud string.*

*Note: If all categories are set to **no**, then only Category 2 will be active.*

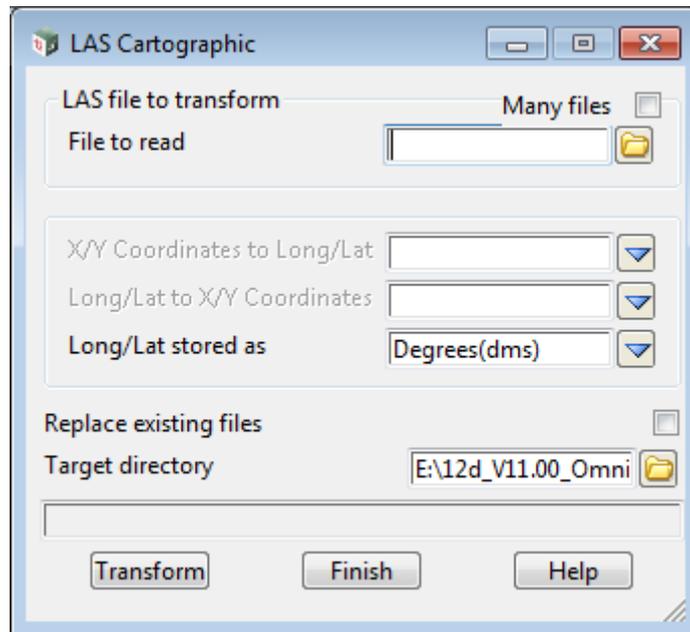
**Button at bottom**

**Read button**

*read the data into the model given in the model field.*

### 15.1.8.3.4 LAS Cartographic

Position of option on menu: File I/O =>Data Input =>Point Clouds =>LAS =>Cartographic transform

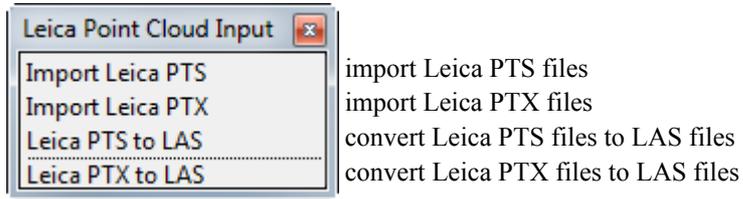


See [15.15.4 LAS Cartographic Transform](#).

## 15.1.8.4 Leica Point Cloud Input

**Position of menu:** File I/O =>Data input =>Point clouds =>Leica

The Leica Point Cloud Input walk-right menu is



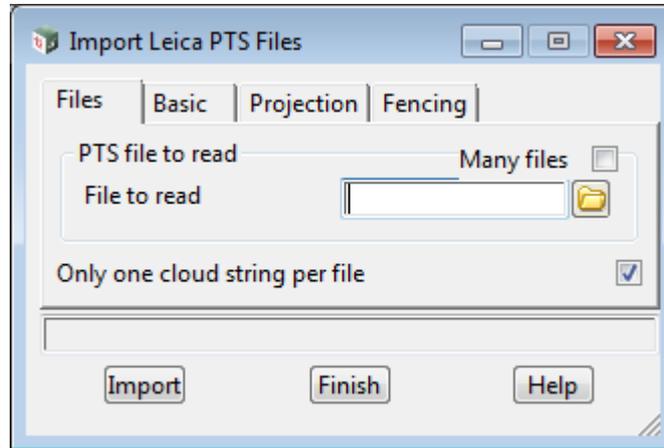
See

<i>Import Leica PTS</i>	<a href="#"><u>15.1.8.4.1 Import Leica PTS Files</u></a>
<i>Import Leica PTX</i>	<a href="#"><u>15.1.8.4.2 Import Leica PTX Files</u></a>
<i>Leica PTS to LAS</i>	<a href="#"><u>15.1.8.4.3 Leica PTS to LAS Files</u></a>
<i>Leica PTX to LAS</i>	<a href="#"><u>15.1.8.4.4 Leica PTX to LAS Files</u></a>

### 15.1.8.4.1 Import Leica PTS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>Leica =>Import Leica PTS

Selecting **Import Leica PTS** brings up the **Import Leica PTS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

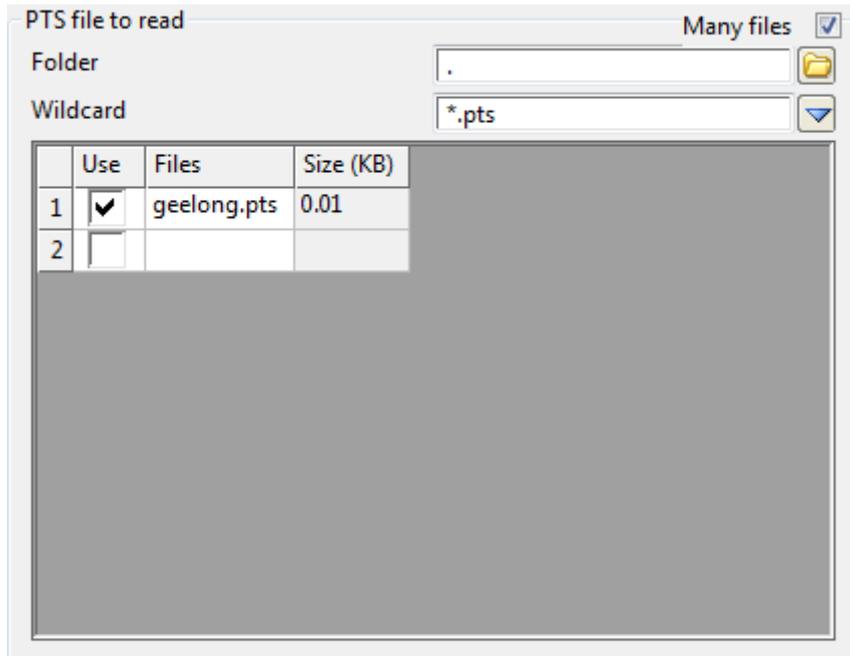
*if **ticked**, a grid to allow multiple pts files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if ticked, read in the file*

**Files** output

*name of the file in the folder*

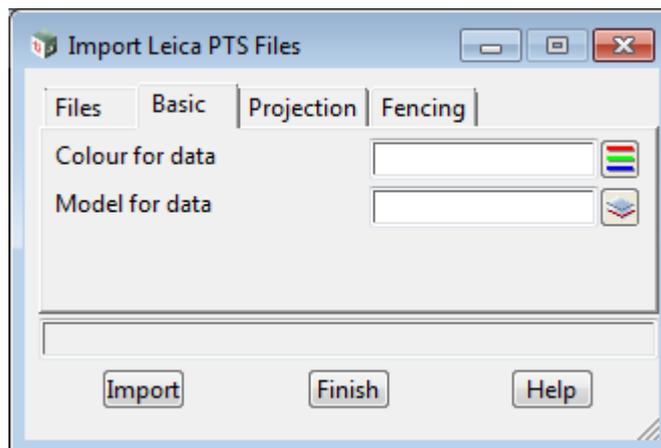
**Size** output

*file size*

**Only one cloud string per file** tick box      ticked

*if ticked, all cloud points will be combined into one 12d cloud string.*

**Basic tab**



**Colour for data** colour box

*string colour to be used if the cloud object does not have colour for each point*

**Model for data** model box



File outside/crossing  
File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

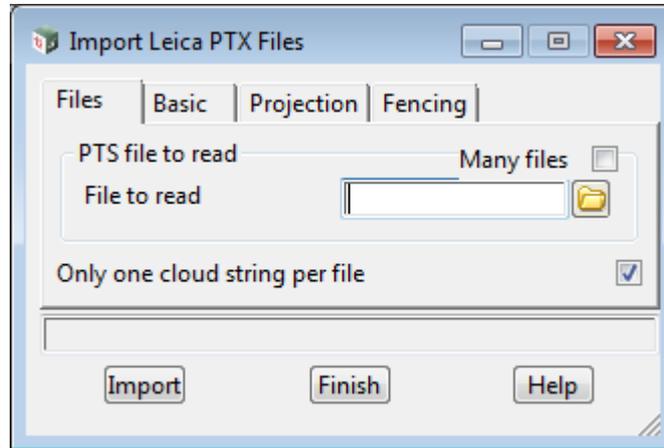
**Import** button

*import the data into the model given in the model field.*

### 15.1.8.4.2 Import Leica PTX Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>Leica =>Import Leica PTX

Selecting **Import Leica PTX** brings up the **Import Leica PTX Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

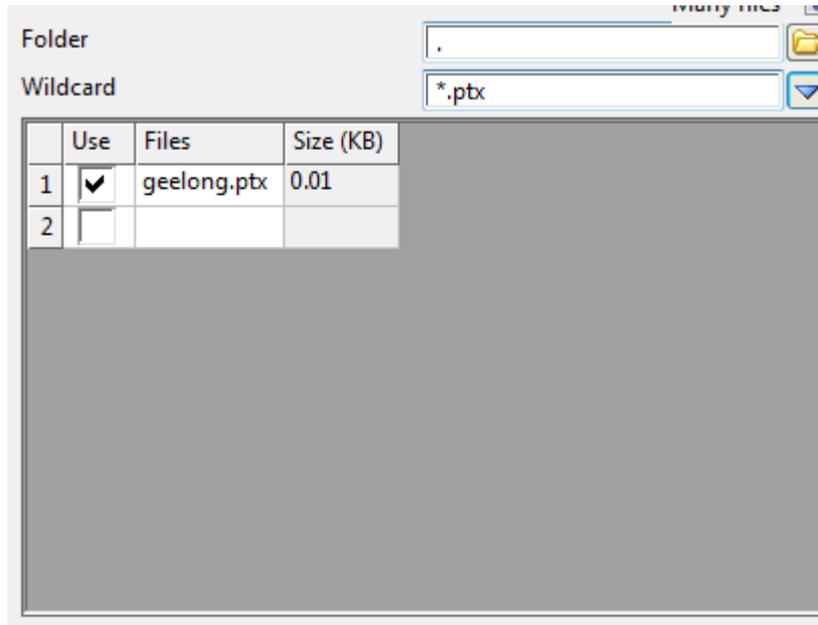
*if **ticked**, a grid to allow multiple ptx files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



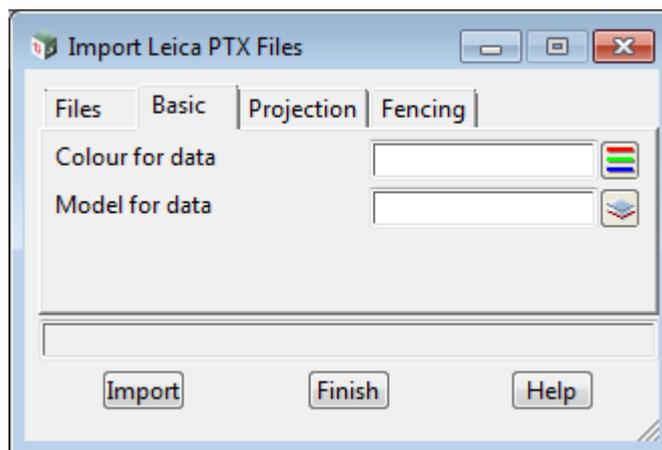
**Use** tick box  
 if *ticked*, read in the file

**Files** output  
 name of the file in the folder

**Size** output  
 file size

**Only one cloud string per file** tick box      *ticked*  
 if *ticked*, all cloud points will be combined into one 12d cloud string.

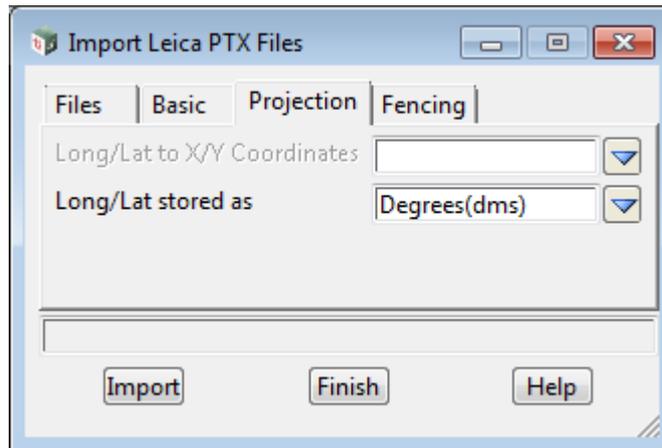
**Basic tab**



**Colour for data** colour box  
 string colour to be used if the cloud object does not have colour for each point

**Model for data** model box  
 name of the model that the data is to be placed in. The model will be created if it does not already exist. This field must be filled in.

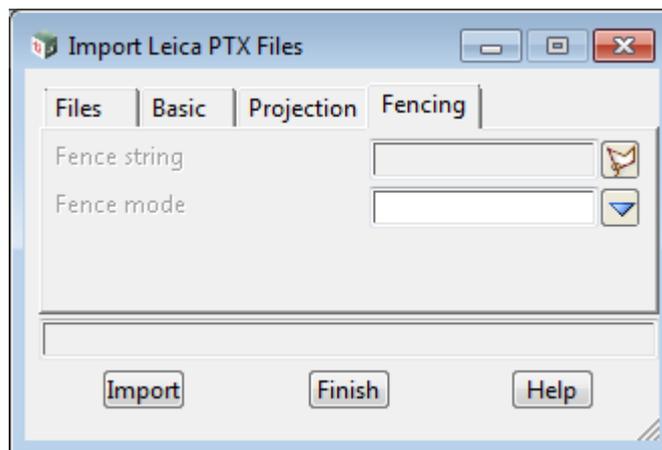
**Projection tab**



**Long/Lat to X/Y Coordinates** choice box available projections  
 if **non-blank**, the cartographic projection to apply to the longitude-latitude values.  
 If **blank**, the co-ordinates are not transformed from (longitude, latitude) to (x,y). Hence the initial (x,y) co-ordinates are transformed to (longitude, latitude) by the transformation given in the **xly co-ordinates to Long/Lat** field and then left in (longitude, latitude). Note that in the southern hemisphere, the latitude values are **negative**.

**Long/Lat stored as** choice box degrees radians degrees decimal degrees  
 format for the longitude and latitudes - either radians, degrees (in [4.17.1 HP Notation](#) for degrees, minutes and seconds) or decimal degrees.

**Fencing tab**



**Fence string** polygon box  
 string to use to restrict the data being read in.

**Fence mode** choice box  
 File inside  
 File inside/crossing  
 File outside  
 File outside/crossing  
 File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

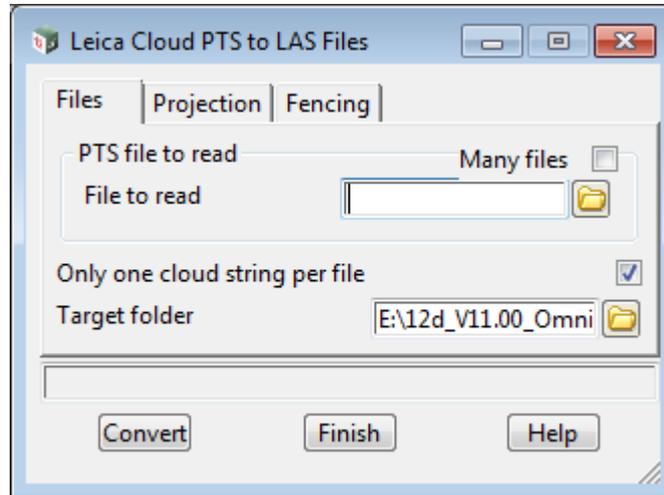
**Import** button

*import the data into the model given in the model field.*

### 15.1.8.4.3 Leica PTS to LAS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>Leica =>Leica PTS to LAS

Selecting Leica PTS to LAS brings up the **Leica Cloud PTS to LAS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

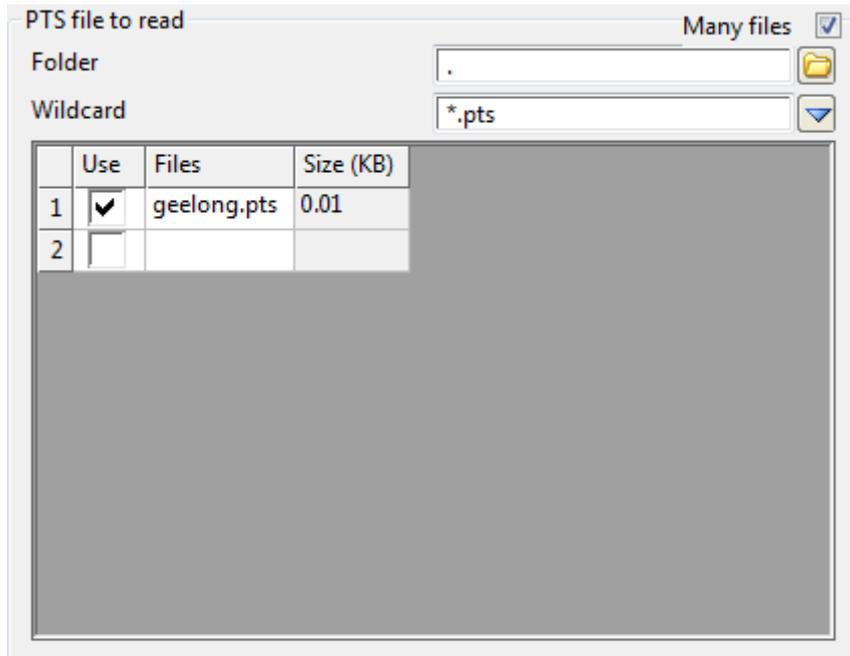
*if **ticked**, a grid to allow multiple files to be converted is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if **ticked**, read in the file*

**Files** output

*name of the file in the folder*

**Size** output

*file size*

**Only one cloud string per file** tick box      **ticked**

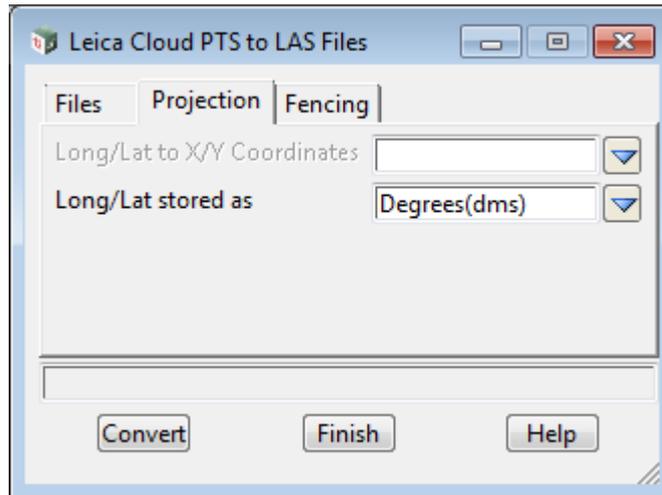
*if **ticked**, all cloud points will be combined into one 12d cloud string.*

**Target folder** file box

*folder where the output LAS file(s) will be created. As it is unknown how many files will be created, it is good practice to specify an empty folder for these files. For example, if the input file is **geelong.pts**, the output file(s) will be **geelong.las** OR **geelong\_1.las**, **geelong\_2.las**, etc.*

*Note: Multiple las files will only be created if the tick box **Only one cloud string per file** is **unticked** and the input file has multiple clouds.*

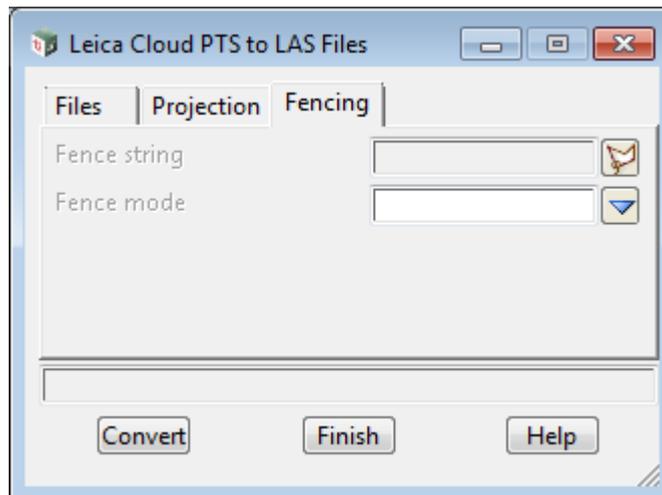
**Projection tab**



**Long/Lat to X/Y Coordinates** choice box available projections  
 if **non-blank**, the cartographic projection to apply to the longitude-latitude values.  
 If **blank**, the co-ordinates are not transformed from (longitude, latitude) to (x,y). Hence the initial (x,y) co-ordinates are transformed to (longitude, latitude) by the transformation given in the **x/y co-ordinates to Long/Lat** field and then left in (longitude, latitude). Note that in the southern hemisphere, the latitude values are **negative**.

**Long/Lat stored as** choice box degrees radians  
 degrees  
 decimal degrees  
 format for the longitude and latitudes - either radians, degrees (in [4.17.1 HP Notation](#) for degrees, minutes and seconds) or decimal degrees.

**Fencing tab**



**Fence string** polygon box  
 string to use to restrict the data being read in.

**Fence mode** choice box  
 File inside  
 File inside/crossing  
 File outside  
 File outside/crossing  
 File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

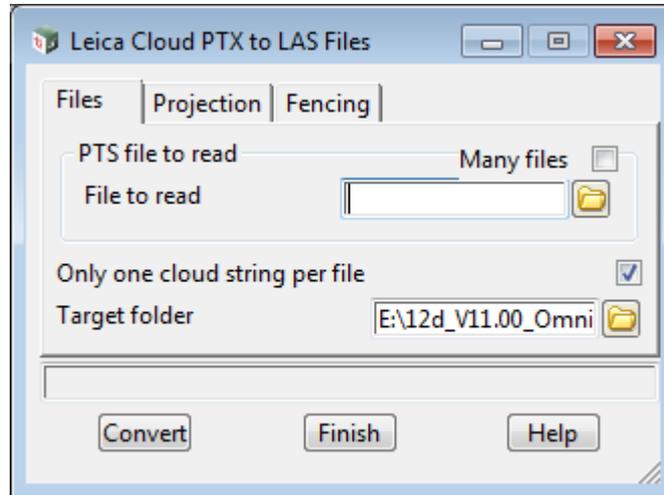
**Convert** button

*convert the data into the model given in the model field.*

### 15.1.8.4.4 Leica PTX to LAS Files

**Position of option on menu:** File I/O =>Data Input =>Point Clouds =>Leica =>Leica PTX to LAS

Selecting Leica PTX to LAS brings up the **Leica Cloud PTX to LAS Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Files tab**

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

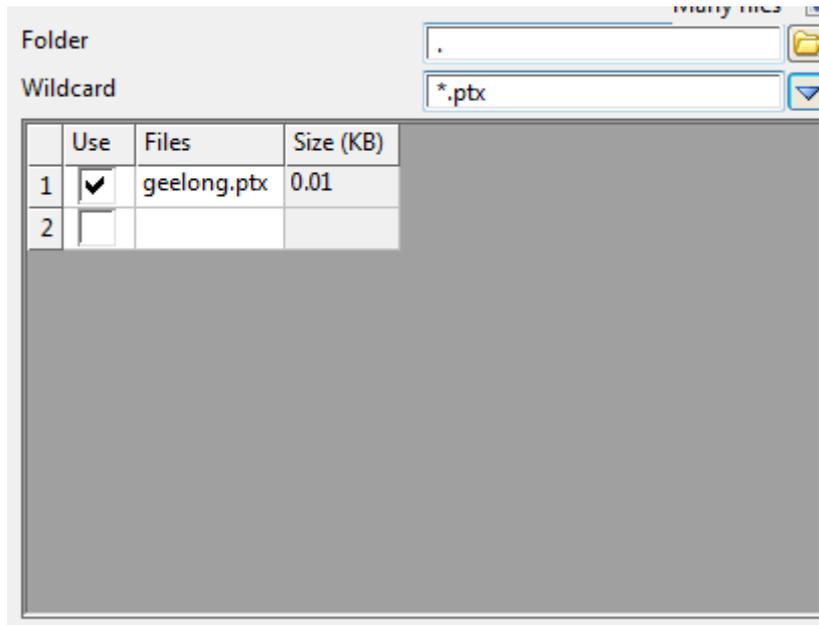
*if **ticked**, a grid to allow multiple files to be converted is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if **ticked**, read in the file*

**Files** output

*name of the file in the folder*

**Size** output

*file size*

**Only one cloud string per file** tick box      **ticked**

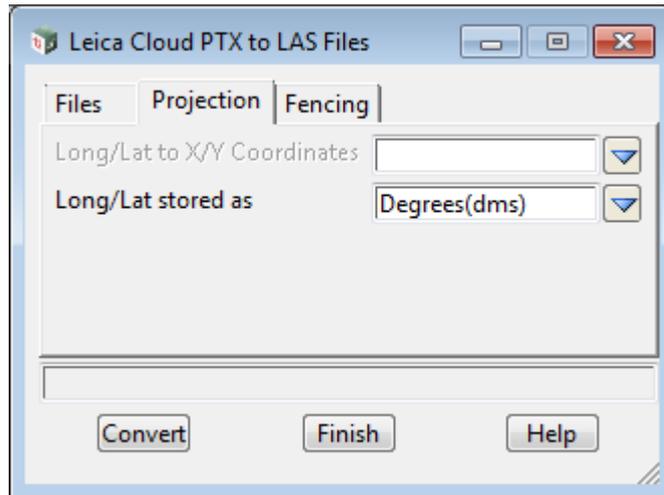
*if **ticked**, all cloud points will be combined into one 12d cloud string.*

**Target folder** file box

*folder where the output LAS file(s) will be created. As it is unknown how many files will be created, it is good practice to specify an empty folder for these files. For example, if the input file is 'geelong.ptx', the output file(s) will be 'geelong.las' OR 'geelong\_1.las', 'geelong\_2.las', etc.*

*Note: Multiple las files will only be created if the tick box 'Only one cloud string per file' is **unticked** and the input file has multiple clouds.*

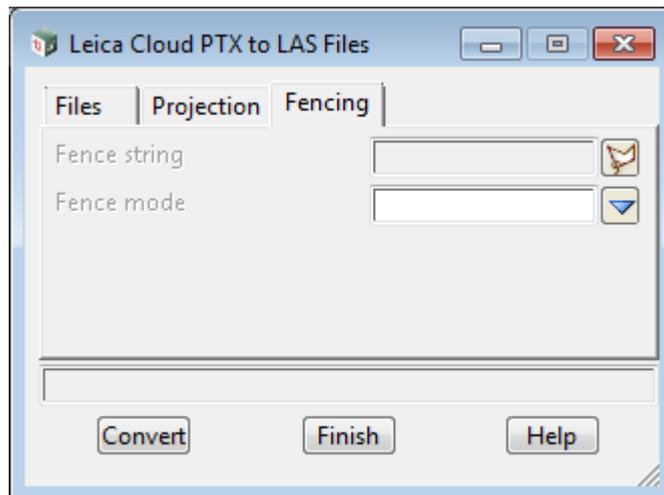
**Projection tab**



**Long/Lat to X/Y Coordinates** choice box available projections  
 if **non-blank**, the cartographic projection to apply to the longitude-latitude values.  
 If **blank**, the co-ordinates are not transformed from (longitude, latitude) to (x,y). Hence the initial (x,y) co-ordinates are transformed to (longitude, latitude) by the transformation given in the **x/y co-ordinates to Long/Lat** field and then left in (longitude, latitude). Note that in the southern hemisphere, the latitude values are **negative**.

**Long/Lat stored as** choice box degrees radians  
 degrees  
 decimal degrees  
 format for the longitude and latitudes - either radians, degrees (in [4.17.1 HP Notation](#) for degrees, minutes and seconds) or decimal degrees.

**Fencing tab**



**Fence string** polygon box  
 string to use to restrict the data being read in.

**Fence mode** choice box  
 File inside  
 File inside/crossing  
 File outside  
 File outside/crossing  
 File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Button at bottom**

**Convert** button

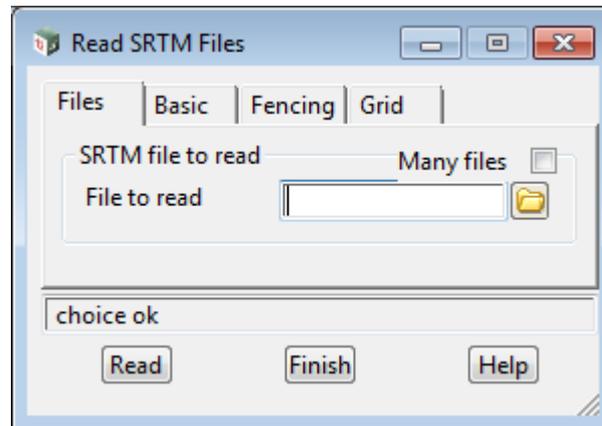
*convert the data into the model given in the model field.*

## 15.1.8.5 Read SRTM Files

**Position of option on menu:** File I/O =>Data Input =>Point clouds =>SRTM

SRTM refers to the Shuttle Radar Topography Mission, an international research effort which obtained a high-resolution digital topographic database of Earth.

Selecting SRTM brings up the **Read SRTM Files** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

### Files tab

<b>Many files</b>	tick box	not ticked	
-------------------	----------	------------	--

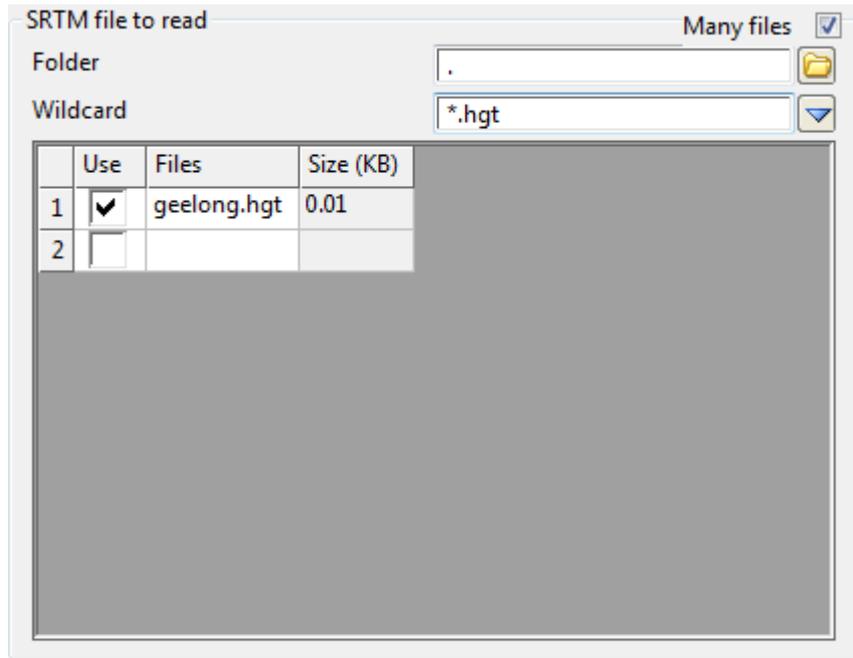
*if **ticked**, a grid to allow multiple hgt files to be read in is opened. A wild card is used to select all the files to be read in.*

<b>Folder</b>	file box		
---------------	----------	--	--

*folder to search for files using the Wild card*

<b>Wildcard</b>	input		
-----------------	-------	--	--

*wild card to use in search for files in the given folder*



**Use** tick box

*if ticked, read in the file*

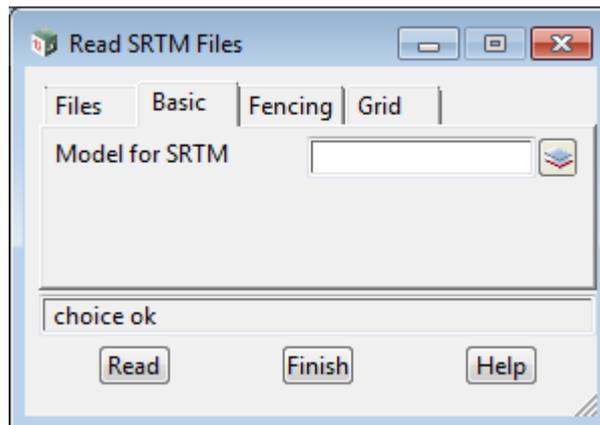
**Files** output

*name of the file in the folder*

**Size** output

*file size*

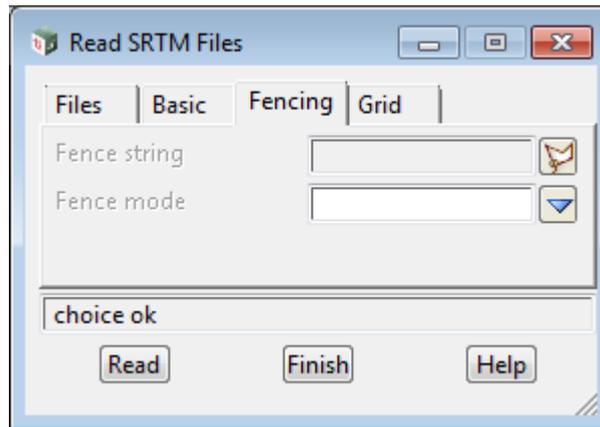
**Basic tab**



**Model for SRTM** model box

*name of the model that the data is to be placed in. The model will be created if it does not already exist. This field must be filled in.*

**Fencing tab**



**Fence string**                      polygon box  
*string to use to restrict the data being read in.*

**Fence mode**                      choice box

- File inside
- File inside/crossing
- File outside
- File outside/crossing
- File crossing

*File inside - read file in if it is totally inside the polygon*

*File inside/crossing - read file in if it is totally inside, or crossing the polygon*

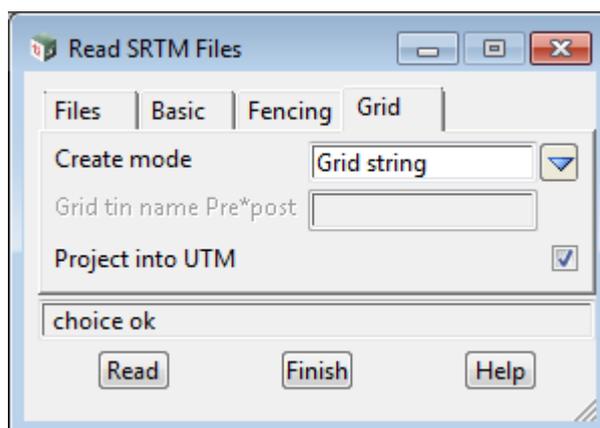
*File outside - read file in if it is totally outside the polygon*

*File outside/crossing - read file in if it is totally outside, or crossing the polygon*

*File crossing - read file in if it is crossing the polygon*

*Note - only whole files are read in.*

**Grid tab**



**Create mode**                      choice box                      Grid string                      Grid string, Grid tin, Strings  
*see [15.10 Grids](#) for an explanation of Grid string and Grid tin*

**Grid tin name Pre\*post**            input  
*only enabled if Grid tin is selected. All grid tins created will have their names derived from the text entered with the \* character beginning at 1 and incrementing for each tin created.*

**Project into UTM**            tick box            ticked

*if ticked, the project will be uploaded into the Universal Transverse Mercator coordinate system.*

**Button at bottom**

**Read**                            button

*read the data into the model given in the model field.*

# 15.2 Data Output

**Position of menu:** File I/O =>Data output

**12d Model** provides output options so that data in a model or on a view can be written out to a disk file. this may be to allow data to be transferred to other programs for further processing or simply to get a readable list of data.

The **Data Output** walk-right menu containing these options is

The image shows two screenshots of software menus. The first is the 'Data Output' menu, which lists various export options. The second is the '12d Output' sub-menu, which is expanded from the '12d' option in the first menu. An arrow points from the '12d' option in the first menu to the '12d Output' sub-menu. To the right of the sub-menu, there are three lines of text explaining the options: 'write out data in 12d Model Archive format', 'write out XML data', and 'write out XML project data'. Below the 'Data Output' menu, there are several lines of text explaining other options: 'a Trimesh can be written out as a 3DFace or a Polyface Mesh', '3D PDF', 'DAE files', 'IFC output', 'Wavefront Obj's', and '3D Printer STL'.

<ul style="list-style-type: none"> <li>Data Output</li> <li>12d</li> <li>ArcView SHP</li> <li>x y z</li> <li>DEMs</li> <li>DGN</li> <li>DWG/DXF/DXB</li> <li>Genio</li> <li>LandXML</li> <li>MapInfo</li> <li>Civilcad V5</li> <li>Geocomp</li> <li>KML</li> <li>TP Stakeout triangles</li> <li>TP Stakeout strings</li> <li>3d PDF</li> <li>DAE</li> <li>IFC</li> <li>OBJ</li> <li>STL</li> <li>Old</li> <li>User</li> </ul>	<ul style="list-style-type: none"> <li>12d Output</li> <li>12d archive data</li> <li>12d XML data</li> <li>12d XML project data</li> </ul>	<p>write out data in <b>12d Model</b> Archive format</p> <p>write out XML data</p> <p>write out XML project data</p>
	<p>a Trimesh can be written out as a 3DFace or a Polyface Mesh</p>	
	<p>3D PDF</p> <p>DAE files</p> <p>IFC output</p> <p>Wavefront Obj's</p> <p>3D Printer STL</p>	

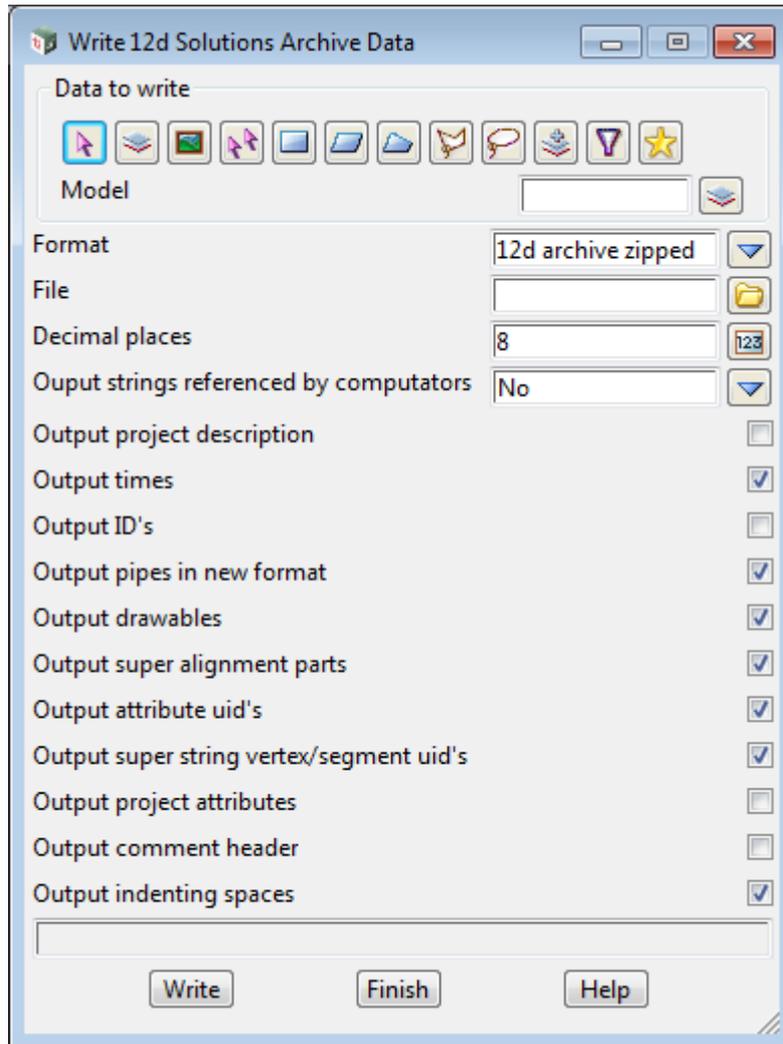
- [15.2.1 12d Archive Output](#)
- [15.2.2 12d XML Data](#)
- [15.2.3 12d XML Project Data](#)
- [15.2.4 Trimesh to DWG](#)
- [15.2.5 Write 3D PDF](#)
- [15.2.6 Export DAE](#)
- [15.2.7 IFC Output](#)
- [15.2.8 Export OBJ](#)
- [15.2.9 Export STL](#)

## 15.2.1 12d Archive Output

Position of option on menu: **File =>Data output =>12d archive data**

The **12d Archive** format is a special format defined by 12d Solutions to allow data to be easily transferred from other programs into 12d Solutions software such as **12d Model**. The **12d Archive** format is given in the Appendix.

Selecting the 12d archive data brings up the **Write 12d Solutions Archive Data** for panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b> <i>data selection type.</i>		Model	
<b>Data source</b> <i>source of data is to be written out to a file.</i>	input		
<b>Format</b>	choice	12d archive zipped	12d archived zipped, 12d archive

use 12d archive zipped (12daz) or 12d archive format (12a). The only difference is that the 12az file is a zipped 12da file with the same name stem.

**File** file box \*.12da or \*.4da

name of the file for the information to be written out to in 12d Archive format. If the file already exists, then you are asked to Append, Replace or Cancel.

**Decimal places** integer box 8

the number of decimal places used when writing the data out.

**Output string references by computators** choice box No Yes, No  
, De-reference

if Yes, any strings referenced by computators by strings in the data source that are not in the data source, are also written out.

If No, any strings referenced by computators by strings in the data source that are not in the data source, are NOT written out.

If De-reference, any super strings in the data source with computators are de-referenced. That is, the computators are replaced by internal elements of the string.

**Output project description** tick box

if ticked, the project description is written out as comments at the top of the file.

**Output times** tick box

if ticked, write out the creation times etc. for the objects.

If not ticked, don't write out the creation times etc. for the objects.

**Output ID's** tick box

if ticked, write out the object IDs.

If not ticked, don't write out the object IDs.

**Output pipes in new format** tick box

if ticked,

If not ticked, don't write out.

**Output drawables** tick box

if ticked, write out the internal super alignment labelling as text.

**Output super alignment parts** tick box

if ticked, write out all the construction details for super alignments.

If not ticked, don't write out the construction details. Just write out the HG and VG segments.

**Output attribute uid's** tick box

if ticked, write out the attribute uids.

**Output super string vertex/segment uid's** tick box

if ticked, write out the any vertex and segment UIDs.

**Output project attributes** tick box

*if ticked, write out the project attributes.*

**Output comment header**    tick box

*if ticked, write out information as comments at the top of the 12da file.*

**Output indenting spaces**    tick box

*if ticked, leave out spaces for indenting lines in the file. This will reduce the size of the text 12da file.*

**Write**                                    button

*write out the data specified by the Data source.*

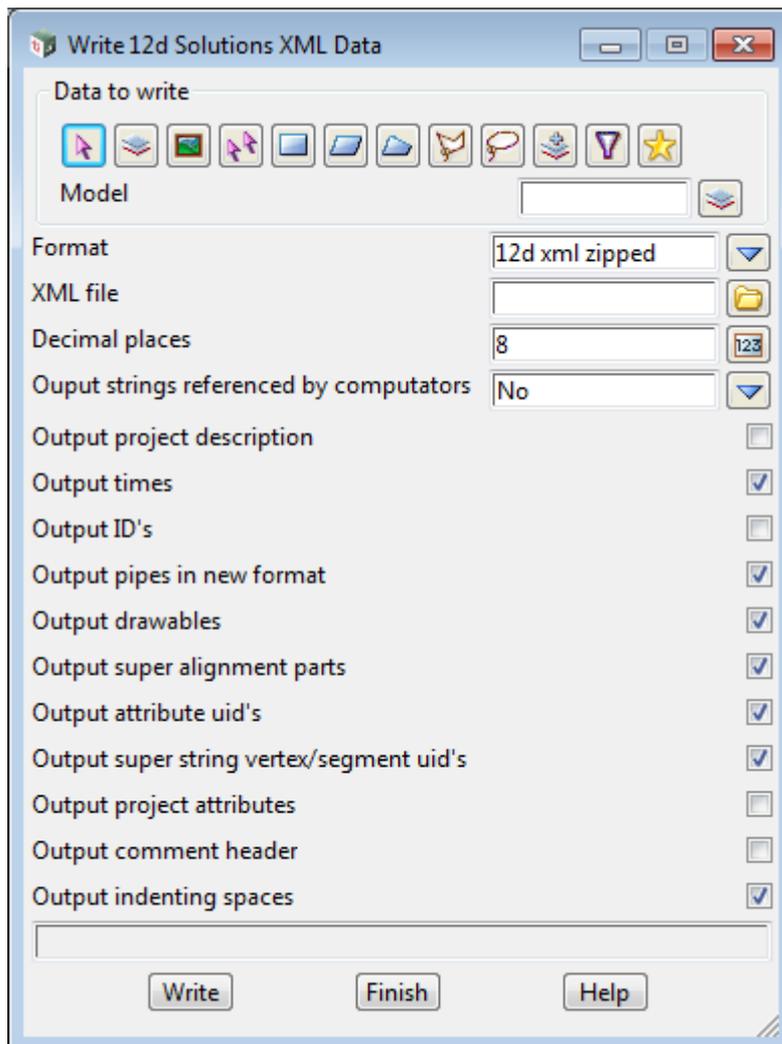
## 15.2.2 12d XML Data

**Position of option on menu:** File I/O =>Data output =>12d =>12d XML data

The **12d XML** format is a special format defined by 12d Solutions to allow data to be easily transferred from other programs into 12d Solutions software such as **12d Model**. The **12d XML** format is given in [28 12d XML File Format](#).

**Note** - 12d XML contains the same information as a 12da file but it is a XML format.

Selecting 12d XML data brings up the **Write 12d Solutions XML Data** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Data source type</b>		Model	
-------------------------	--	-------	--

*Data to search - for a full description go to [4.19.3 Data Source](#).*

<b>Data source</b>	input		
	<i>source of data to be written out to a file.</i>		

<b>Format</b>	choice	12d xml zipped	12d xml zipped, 12d xml
---------------	--------	----------------	----------------------------

*use 12d xml zipped (12dxmlz) or 12d xml format (12xml). The only difference is that the 12xmlz file is a zipped 12dxml file with the same name stem.*

**XML file** file box

*name of the file for the information to be written out to in 12d XML format. If the file already exists, then you are asked to Append, Replace or Cancel.*

**Decimal places** integer box 8

*the number of decimal places used when writing out the data.*

**Output string references by computators** choice box No Yes, No  
, De-reference

*if Yes, any strings referenced by computators by strings in the data source that are not in the data source, are also written out.*

*If No, any strings referenced by computators by strings in the data source that are not in the data source, are NOT written out.*

*If De-reference, any super strings in the data source with computators are de-referenced. That is, the computators are replaced by internal elements of the string.*

**Output project description** tick box

*if ticked, the project description is written out as comments at the top of the file.*

**Output times** tick box

*if ticked, write out the creation times, etc. for the objects.*

*If not ticked, don't write out the creation times etc. for the objects.*

**Output ID's** tick box

*if ticked, write out the object IDs.*

*If not ticked, don't write out the object IDs.*

**Output pipes in new format** tick box

*if ticked,*

*If not ticked, don't write out.*

**Output drawables** tick box

*if ticked, write out the internal super alignment labelling as text.*

**Output super alignment parts** tick box

*if ticked, write out all the construction details for super alignments.*

*If not ticked, don't write out the construction details. Just write out the HG and VG segments.*

**Output attribute uid's** tick box

*if ticked, write out the attribute uids.*

**Output super string vertex/segment uid's** tick box

*if ticked, write out the any vertex and segment UIDs.*

**Output project attributes** tick box

*if ticked, write out the project attributes.*

**Output comment header** tick box

*if ticked, write out information as comments at the top of the 12da file.*

**Output indenting spaces** tick box

*if ticked, leave out spaces for indenting lines in the file. This will reduce the size of the text 12d XML file.*

**Write** button  
*write out the data specified by the Data source.*

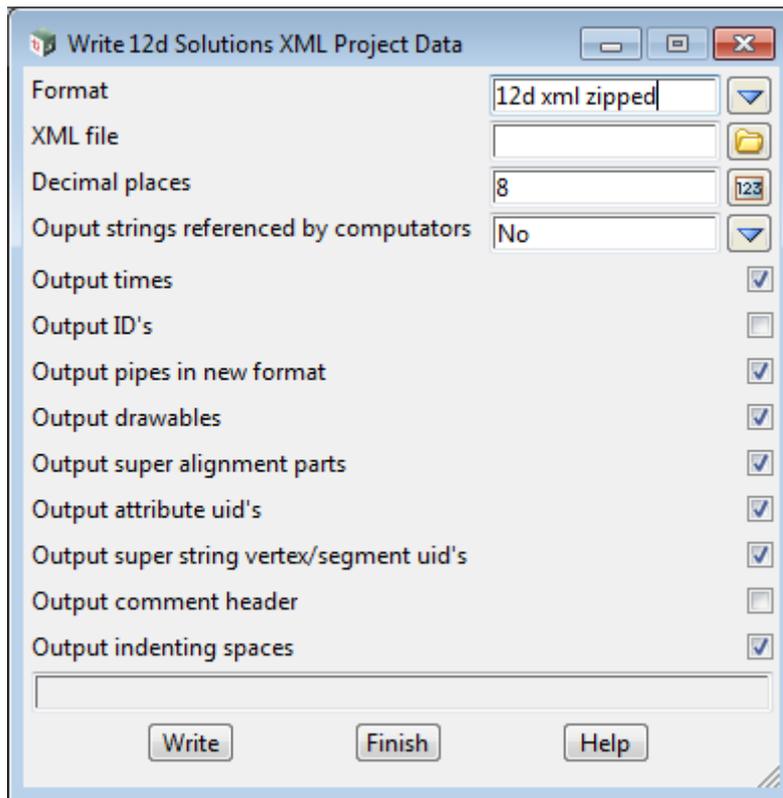
## 15.2.3 12d XML Project Data

**Position of option on menu:** File I/O =>Data output =>12d =>12d XML project data

The **12d XML** format is a special format defined by 12d Solutions to allow data to be easily transferred from other programs into 12d Solutions software such as **12d Model**. The **12d XML** format is given in the Appendix.

**Note** - 12d XML contains the same information as a 12da file but it is a XML format.

Selecting 12d XML project data brings up the **Write 12d Solutions XML Project Data** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>XML file</b>	folder box		
-----------------	------------	--	--

*name of the file for the information to be written out to in 12d XML format. If the file already exists, then you are asked to Append, Replace or Cancel.*

<b>Decimal places</b>	integer box	8	
-----------------------	-------------	---	--

*the number of decimal places used when writing out the data.*

<b>Output strings referenced by computators</b>	choice box	No	Yes, No , De-reference
---	------------	----	---------------------------

*if Yes, any strings referenced by computators by strings in the data source that are not in the data source, are also written out.*

*If No, any strings referenced by computators by strings in the data source that are not in the data source, are NOT written out.*

*If De-reference, any super strings in the data source with computators are de-referenced. That is, the*

*computators are replaced by internal elements of the string.*

**Output times**                      tick box

*if ticked, write out the creation times, etc. for the objects.*

*If not ticked, don't write out the creation times etc. for the objects.*

**Output ID's**                      tick box

*if ticked, write out the object IDs.*

*If not ticked, don't write out the object IDs.*

**Output pipes in new format**              tick box

*if ticked,*

*If not ticked, don't write out.*

**Output drawables**              tick box

*if ticked, write out the internal super alignment labelling as text.*

**Output super alignment parts**    tick box

*if ticked, write out all the construction details for super alignments.*

*If not ticked, don't write out the construction details. Just write out the HG and VG segments.*

**Output attribute uid's**              tick box

*if ticked, write out the attribute uids.*

**Output super string vertex/segment uid's**              tick box

*if ticked, write out the any vertex and segment UIDs.*

**Output comment header**    tick box

*if ticked, write out information as comments at the top of the 12da file.*

**Output indenting spaces**    tick box

*if ticked, leave out spaces for indenting lines in the file. This will reduce the size of the text 12d XML file.*

**Write**                                  button

*write out the data specified by the Data source.*

## 15.2.4 Trimesh to DWG

Trimeshes can be written out to a DWG file as either Polyface Meshes or 3DFaces or not at all.

The choice is set by the environment variable TRIMESH\_TO\_DWG\_MODE\_4D in the **External Apps >AutoCAD** node of the **Edit Environment Variables** panel.

An AutoCAD Polyface Mesh can only have one colour so if the choice is **Polyface mesh** then the trimesh colour is used and any differently coloured trimesh triangles will only go out in the trimesh colour.

If the choice is **Faces** then each 3DFace is given the colour of the corresponding triangle of the trimesh.

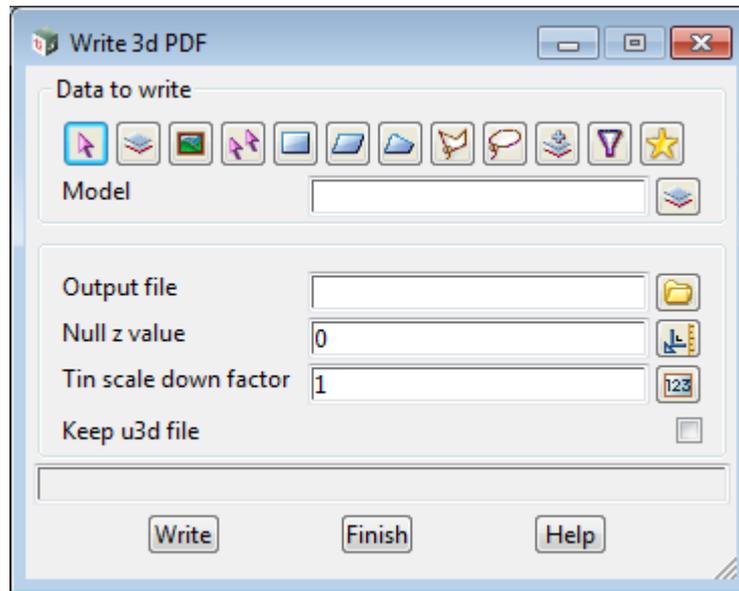
## 15.2.5 Write 3D PDF

This option writes out a Data Source as a 3D pdf file.

Currently no textures are applied.

Position of option on menu: **File =>Data output =>Export 3d pdf**

Selecting **Export 3d pdf** brings up the **Write 3d pdf** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b> <i>data selection type.</i>		Model	
<b>Data source</b> <i>source of data is to be written out to a 3d pdf file.</i>	input		
<b>Output file</b> <i>name of the 3d pdf that is created.</i>	file box		*.pdf
<b>Null z value</b> <i>z-value to use in place of null z-values in 12d strings.</i>	real box		
<b>Tin scale down factor</b> <i>if this value is greater than one then triangles from tins start to be dropped out.</i>	integer box		
<b>Keep u3d file</b> <i>if ticked, the u3d file that becomes part of the 3d PDF is also kept as a separate file.</i>	tick box		
<b>Write</b> <i>write out the specified data to the pdf file.</i>	button		

To view the 3D PDF, see [15.2.5.1 How to View the Generated 3D PDF](#)

### 15.2.5.1 How to View the Generated 3D PDF

At the moment the option is not setting an eye and target point do to look at the data in the generated 3d pdf file, follow these steps:

1. Open the 3d pdf file with Acrobat pdf viewer. You'll then get a bit of a box drawn on the screen.
2. Move the cursor into the box and you will see the message **Click to activate**. Then click anywhere inside the box.
3. Then Right click to bring up a menu and choose **Full Screen Multimedia** from the menu. You should then see your data fitted in the view and displaying.
4. Hold down the left mouse button to rotate. Wheel to zoom in and out. Shift LB to pan.

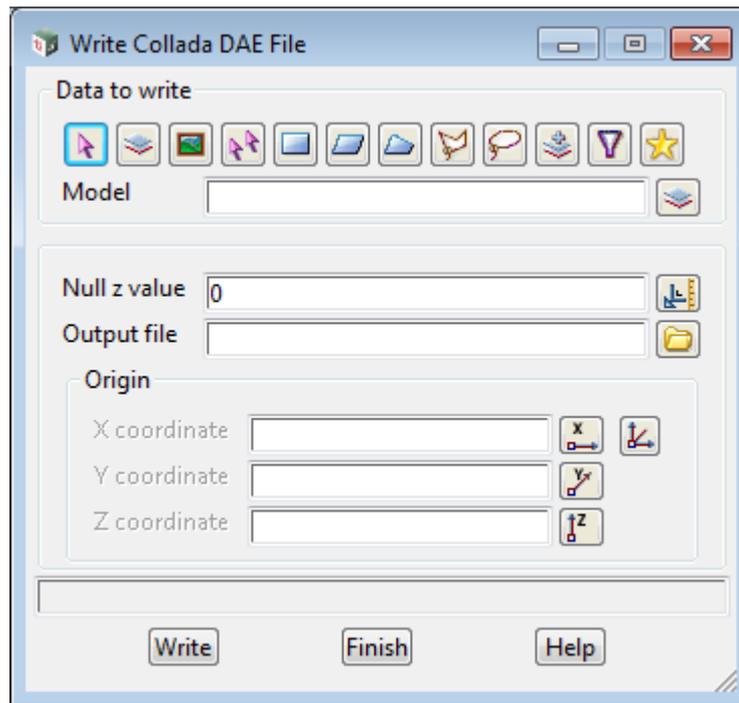
## 15.2.6 Export DAE

**Position of option on menu:** File I/O =>Data output => DAE

The Collada Digital Asset Exchange (DAE) file format is a published format for writing out 3d shapes for interchange between various graphics packages.

Trimeshes, extrusions, pipes and drainage strings can be written to a DAE file.

Selecting DAE brings up the **Write Collada DAE File** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of data to be written out to a DAE file.</i>			
<b>Null z value</b>	real box	0	measures menu
<i>value to use in place of any null z values.</i>			
<b>Output file</b>	file box		folder browse
<i>name of the file to write the DAE information to.</i>			
<b>Origin</b>	point select box		
<i>.</i>			
<b>Write</b>	button		
<i>write out the DAE file.</i>			

## 15.2.7 IFC Output

**Position of menu:** File I/O =>Data output =>IFC

The **IFC Output** options write out **12d Model** data in the IFC STEP format, version **IFC 2x3**.

The **Industry Foundation Classes** (IFC) data model is intended to describe building and construction industry data.

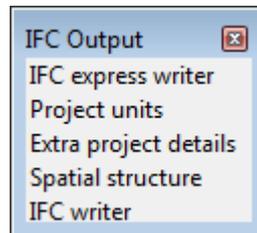
It is a platform neutral, open file format specification that is not controlled by a single vendor or group of vendors.

It is an object-based file format with a data model developed by **buildingSMART** (formerly the International Alliance for Interoperability, IAI) to facilitate interoperability in the architecture, engineering and construction (AEC) industry, and is a commonly used collaboration format in **Building Information Modeling** (BIM) based projects. See <http://www.buildingsmart-tech.org>

For more information on IFC's, see [11.6 IFCs](#).

The data in **12d Model** that can be written out to IFC's includes super strings, TINs, drainage strings and trimeshes.

The **IFC Output** walk-right menu is



For the options:

- |                              |  |
|------------------------------|--|
| <i>IFC express writer</i>    | <a href="#">15.2.7.1 IFC 2x3 Express Writer</a>                    |
| <i>Project units</i>         | <a href="#">15.2.7.2 Project Units</a>                             |
| <i>Extra project details</i> | <a href="#">15.2.7.3 Extra Project Details</a>                     |
| <i>Special tag structure</i> | <a href="#">15.2.7.4 Defining and Assigning Spatial Structures</a> |
| <i>IFC writer</i>            | <a href="#">15.2.7.5 IFC Writer</a>                                |

For general information on IFCs, see [11.6 IFCs](#).

For information on what **12d Model** objects are written out to IFCs, and how they are written, see [11.6.3 Representation of 12d Objects as IFCs](#).

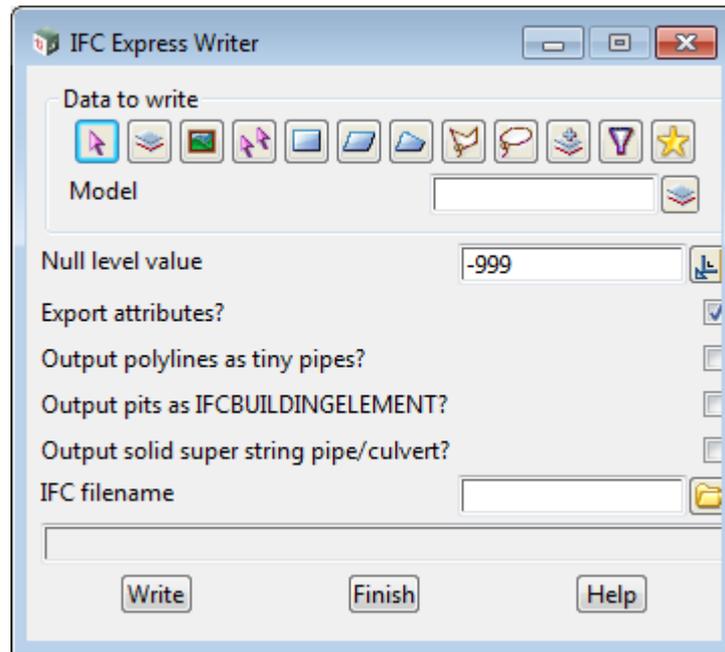
### 15.2.7.1 IFC 2x3 Express Writer

**Position of option on menu:** File I/O =>Data output =>IFC => IFC express writer

The IFC express writer option writes out a data source of objects to a file in the IFC 2x3 format (see [35.6.2 Some IFC 2x3 Definitions](#)).

All the data is placed in the one IFC spatial structure IFCBUILDING. See [15.2.7.4 Defining and Assigning Spatial Structures](#).

Selecting IFC express writer brings up the IFC Express Writer panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	

*Data to search - for a full description go to [4.19.3 Data Source](#).*

<b>Data source</b>	input		
	<i>source of data to be written out to the ifc file.</i>		

<b>Export attributes</b>	tick box	ticked	
	<i>if <b>ticked</b>, the 12d Project Attributes will be exported to the IFC file to the ifcProject. For 12d Objects that are represented as ifcBuildingElementProxy elements, the string/tin/trimesh attributes are exported as an ifcPropertySet.</i>		

<b>Export polylines as tiny pipes ?</b>	tick box	not ticked	
	<i>if <b>ticked</b>, super strings without an actual diameter or culvert, are written out with a small diameter.</i>		
	<i>This is there because some software will need not read in ifc's of strings with no diameter.</i>		

<b>Export pits as IFC_BUILDINGELEMENT ?</b>	tick box	not ticked	
	<i>if <b>ticked</b>, drainage pits and sewer maintenance holes are written out a IfcBuildingElement instead of IfcFlowStorageDevice.</i>		
	<i>This is for writing out ifc's for NavisWorks because NavisWorks does not support</i>		

*IfcFlowStorageDevice*. See [35.6.2.5 ifcBuildingElement](#) and [35.6.2.7 ifcFlowStorageDevice](#).

**Export solid super string pipe/culvert?** tick box not ticked

*if ticked, super string pipes and culverts are written out solid IFC elements.*

*If not ticked, super string pipes and culverts are written out hollow IFC elements.*

### Filename

**IFC Filename**

file box

available \*.ifc files

*the name of the IFC file to write the data source to.*

### Buttons at bottom

**Write File**

button

*write the IFC 2x3 file.*

Continue to next section [15.2.7.2 Project Units](#) or go back to [15.2.7 IFC Output](#).

## 15.2.7.2 Project Units

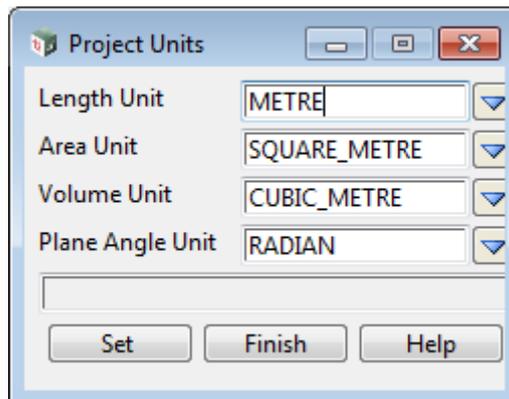
**Position of option on menu:** File I/O =>Data output =>IFC => Project units

This option creates the Project attributes group **ifc\_attributes/ifc\_project\_units** and the values of the attributes are written out when an IFC STEP file (IFC file) is created.

**Note** that this does not change the data stored inside **12d Model** or written out to the IFC file.

The user must set the **IFC Units** to match what units they are been using inside **12d Model**. And these by default are METRE, SQUARE\_METRE, CUBIC\_METRE and RADIAN.

Selecting **Project units** brings up the **Project Units** panel:



and if the Project attributes group **ifc\_attributes/ifc\_project\_units** exists, the values are displayed in the panel. If the Project attributes group **ifc\_attributes/ifc\_project\_units** does not exist, then it is created when the **Set** button is pressed.

The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Length Unit</b>	choice box	as per Project Attributes	MILLIMETRE METRE, KILOMETRE
<i>length units used globally in the IFC file. Defined as SI units. This must match what you units data is in within 12d Model.</i>			
<b>Area Unit</b>	choice box	as per Project Attributes	SQUARE)MILLIMETRE, SQUARE_METRE, SQUARE_KILOMETRE
<i>area units used globally in the IFC file. Defined as SI units. This must match what you units data is in within 12d Model</i>			
<b>Volume Unit</b>	choice box	as per Project Attributes	CUBIC_MILLIMETRE, CUBIC_METRE, CUBIC_KILOMETRE
<i>volume units used globally in the IFC file. Defined as SI units. This must match what you units data is in within 12d Model</i>			
<b>Plane Angle Unit</b>	choice box	as per Project Attributes	RADIAN, DEGREES
<i>plane angle units used globally in the IFC file. Defined as SI units.</i>			
<b>Set</b>	button		
<i>updates/creates the Project attribute group ifc_attributes/ifc_project_units with the values in the</i>			

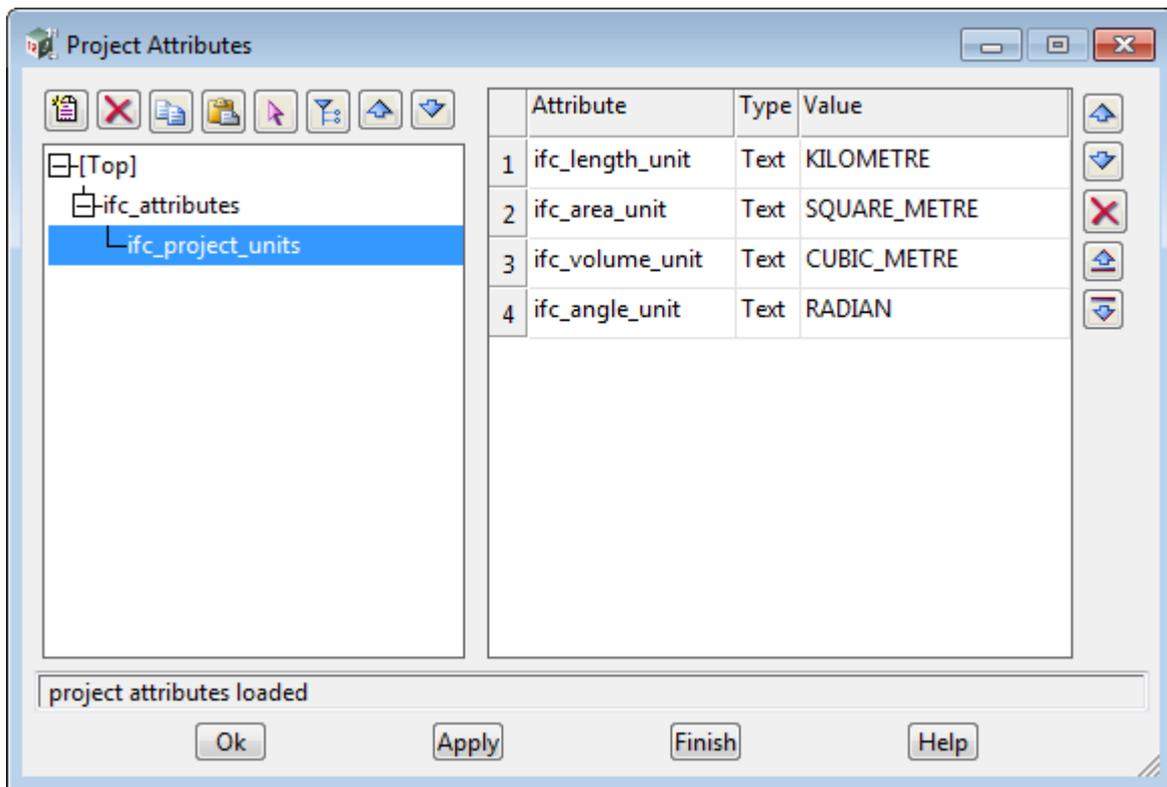
panel.

**Note**

The attributes can be displayed (and modified) using the option

**Project =>Utilities =>Attributes**

which brings up panel **Project Attributes**.



**Warning:**

Be careful making any changes using **Project Attributes**. The attribute name can't be and the attribute values must be valid IFC values.

Continue to next section [15.2.7.3 Extra Project Details](#) or go back to [15.2.7 IFC Output](#).

### 15.2.7.3 Extra Project Details

**Position of option on menu:** File I/O =>Data output =>IFC => Extra project details

This option

- (a) defines the path name of the **Spatial Schema File** that defines the types of IFC spatial elements that 12d elements can be tagged as, and the relationship between the allowed IFC spatial elements.

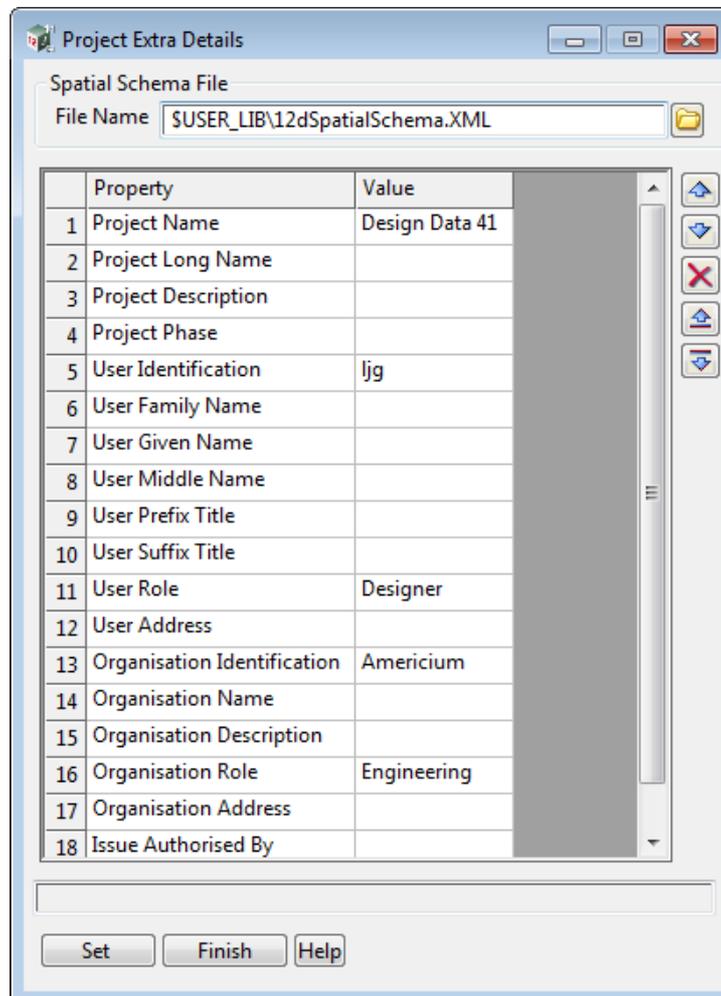
and

- (b) creates and updates the **Project Attributes** group

***ProjectExtraDetails.***

that collects information to be written out as part of **ifcProject**.

Selecting **Extra project details** displays the panel **Project Extra Details**:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Spatial Schema File**

<b>File Name</b>	file box	available *.xml files
------------------	----------	-----------------------

*name of the XML file which will be used by the **Spatial tag structure** option (the **Spatial Tag***

**Aggregation panel** (the next step before using the IFC Writer to create an IFC file). For more information on when and how this file is created, see [15.2.7.3.1 Spatial Schema File](#).

## Grid

### Property/Value

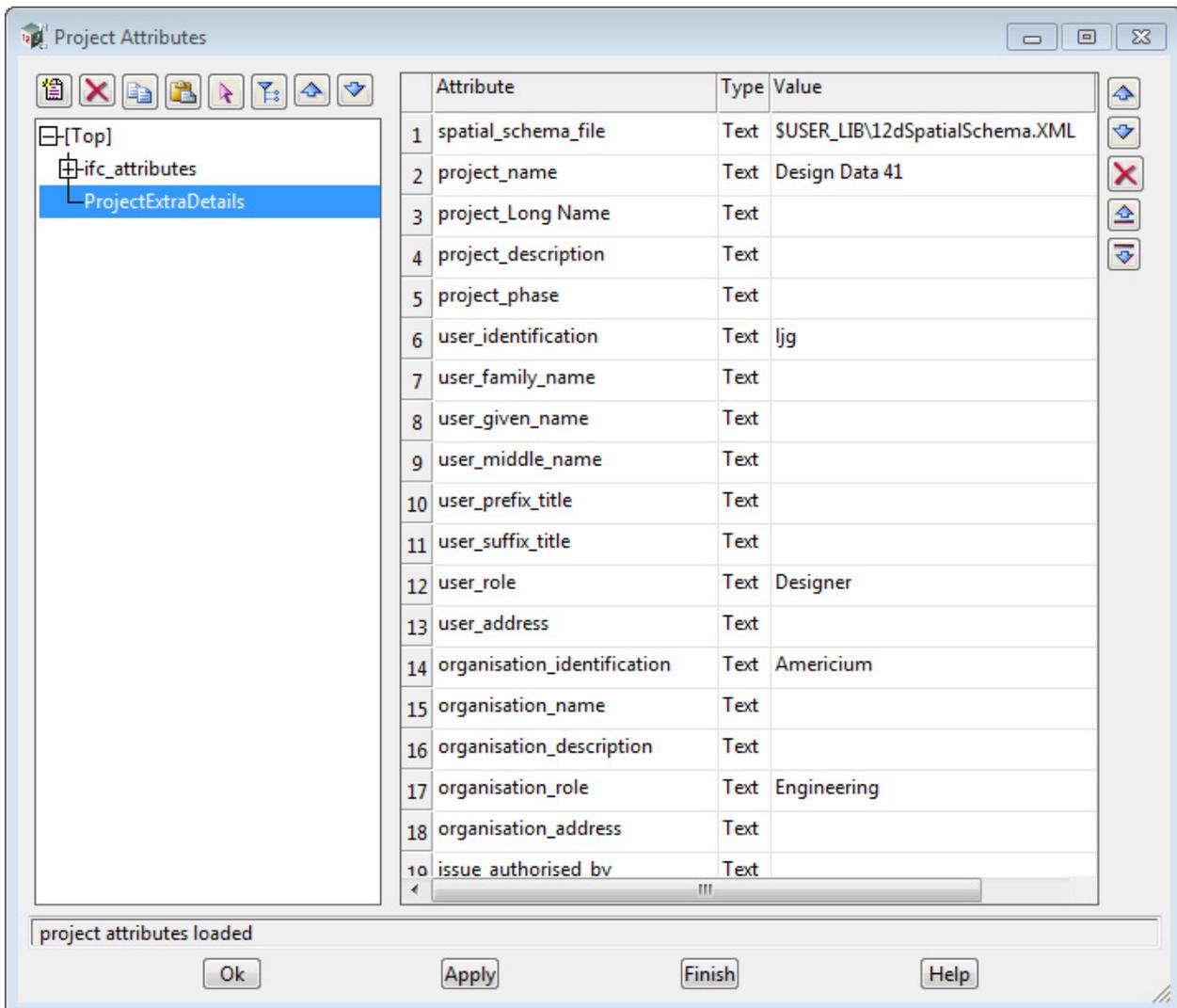
this grid allows the user to enter extra details about the IFC project and the information is stored as Project attributes in the group **ProjectExtraDetails**. The values of these attributes are written out to the IFC file as part of the **ifcProject** element.

### Buttons at bottom

#### Set button

set the value of the attribute **spatial\_schema\_file** in the **ProjectExtraDetails** group to the name given in the File name field.

Then use the values in the Property/Value Grid to create/update the values of the attributes in the Project attributes group **ProjectExtraDetails**. Note that these attributes are written out to the **ifcProject** element in any IFC file that is created by the **IFC writer** option (see [15.2.7.5 IFC Writer](#)).



Continue to the next sub-section [15.2.7.3.1 Spatial Schema File](#) or the next section [15.2.7.4 Defining and Assigning Spatial Structures](#), or return to [15.2.7 IFC Output](#).

### 15.2.7.3.1 Spatial Schema File

The **Spatial Schema File** is an XML file which is used by the **Spatial tag structure** option (the **Spatial Tag Aggregation** panel) to define types of IFC spatial elements, and the allowed hierarchical relationship between the types.

For example, the type **Site** (ifcSite) is allowed as a standalone entity.

The type **Site Partial** (ifcBuilding) can't be standalone but must be associated with a specific **Site** and only a **Site**. Whereas **Building** (ifcBuilding) also can't stand alone but it can be associated with a specific **Site** or a specific **Site Partial**.

When **Extra project details** is selected the Project attribute:

*ProjectExtraDetails/spatial\_schema\_file*

is searched for and

- (a) if the **attribute exists** then a file of the name of the value of the attribute is searched for.

If the file does not exist, then a **Question?** panel comes up:



If **OK** is pressed then a file of the given name is created and a default spatial schema written to the file.

OR

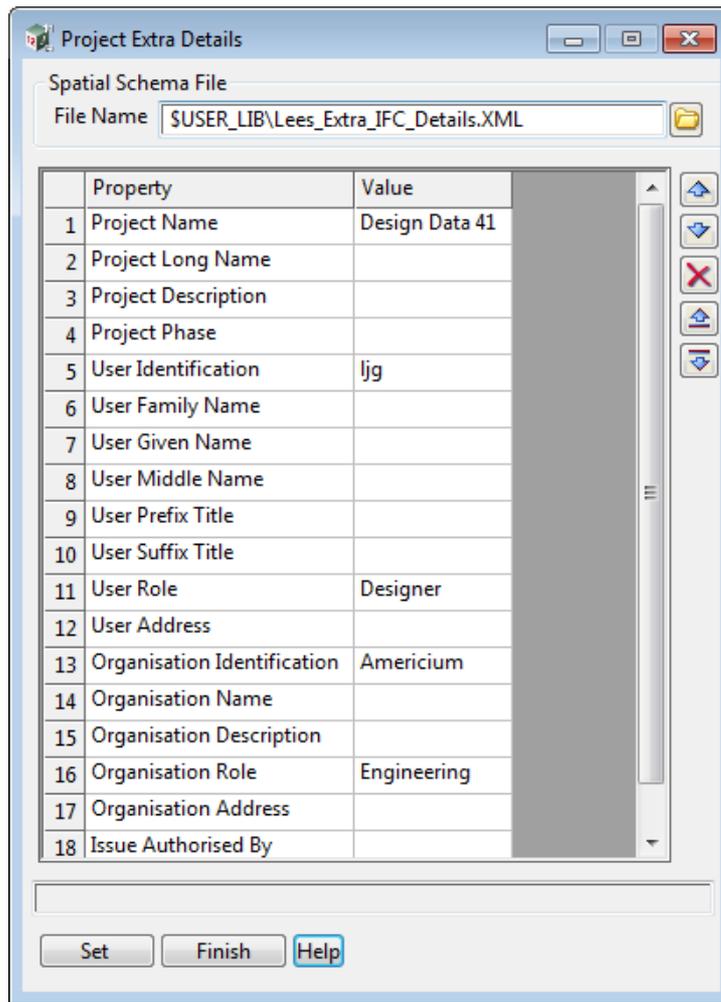
- (b) if the **attribute doesn't exist** then a file of the name called *12dSpatialSchema.XML* is searched for in \$USER\_LIB.

If the file does not exist, then a **Question?** panel comes up:



If **OK** is pressed then the file \$USER\_LIB\12dSpatialSchema.XML is created and a default spatial schema written to the file.

The panel **Project Extra Details** is then brought up displaying name of the *Spatial Schema File*, and also the grid of parameters that are stored in the Project attribute group *ProjectExtraDetails* for writing out to the *ifcProject* element in the IFC file.



Continue to next section [15.2.7.4 Defining and Assigning Spatial Structures](#) or return to [15.2.7 IFC Output](#).

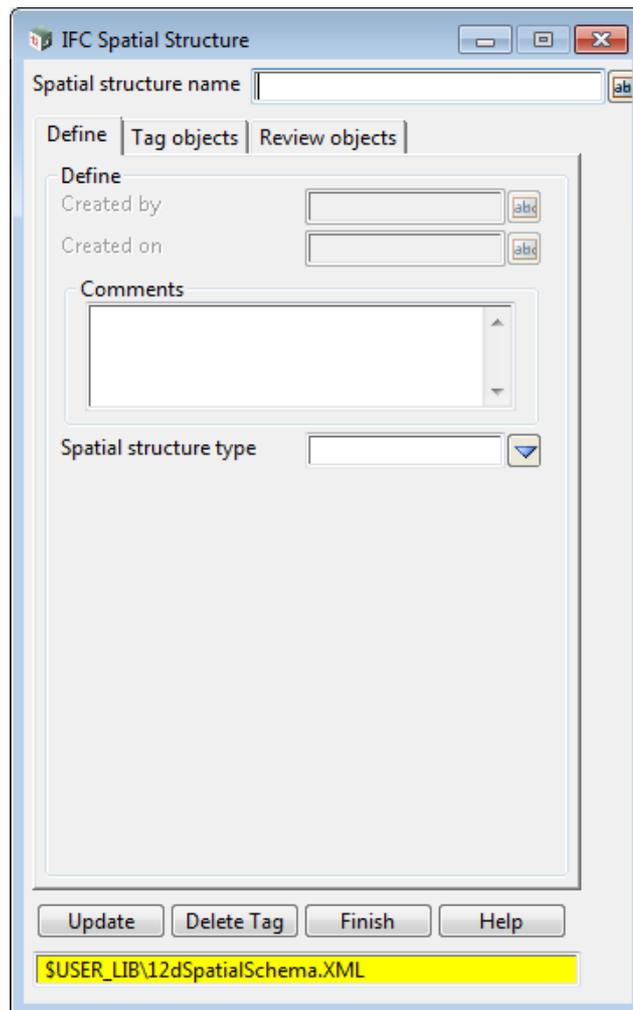
### 15.2.7.4 Defining and Assigning Spatial Structures

**Position of option on menu:** File I/O =>Data output =>IFC => Spatial structures

This option actually has three functions:

1. It creates the named structures as tags using the *Spatial Schema File* referred to by the attribute **spatial\_schema\_file** in the Project attribute group **ProjectExtraDetails** (see [15.2.7.4 Defining and Assigning Spatial Structures](#)).
2. it tags models and/or strings with a spatial structure defined in 1.
3. it select all items of given spatial structure and optionally remove the spatial structure tag from any of them.

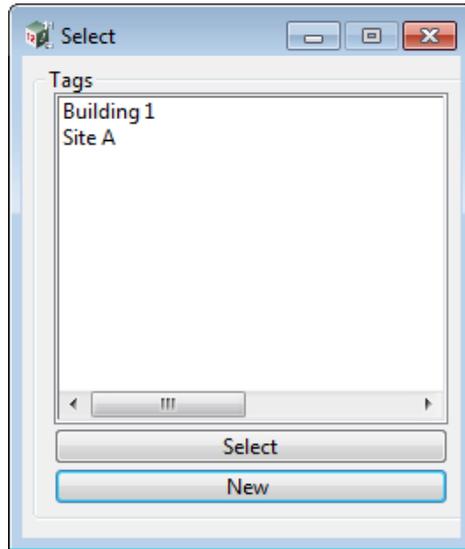
Selecting **Spatial structure** brings up the **IFC Spatial Structure** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Spatial structure name</b>	input		Select/New Spatial structures

*click inside the text entry field bring up the **Select Spatial Structure** panel.*



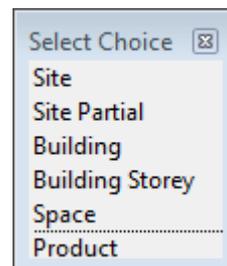
If **New** is selected then **Create Spatial Structure** panel is brought up to.

If a **Spatial Structure** is double clicked on in the list, or a **Spatial Structure** highlighted in the list and **Select** pressed, then the rest of the panel is filled with the information for the selected **Spatial Structure**. the information can then be edited and then updated by clicking on the **Update** button.

### Define Tab

this tab is used when a new **Spatial Structure** is being created or an existing one edited.

<b>Created by</b>	output	from selected <b>Spatial Structure</b> cannot be edited from this tab.
<b>Created on</b>	output	from selected <b>Spatial Structure</b> cannot be edited from this tab.
<b>Comments</b>	input/output	from selected <b>Spatial Structure</b> displays the existing comments which can be edited and extra comments added.
<b>Spatial structure type</b>	choice box	from selected <b>Spatial Structure</b>



displays the spatial structure types for the selected **Spatial Structure**. This can be changed and the choices are defined in the spatial schema file. See [15.2.7.3.1 Spatial Schema File](#).

if **Spatial structure** type is set to **Site**, the following fields are displayed.

The fields and names on the left hand side of the panel are populated from the Spatial Schema File (see [15.2.7.3.1 Spatial Schema File](#)). The values for the fields can be created/edited and when the **Update** button is pressed, the values are stored as a subgroup of the Tags group of the Project attributes. The subgroup has the name of the Site.

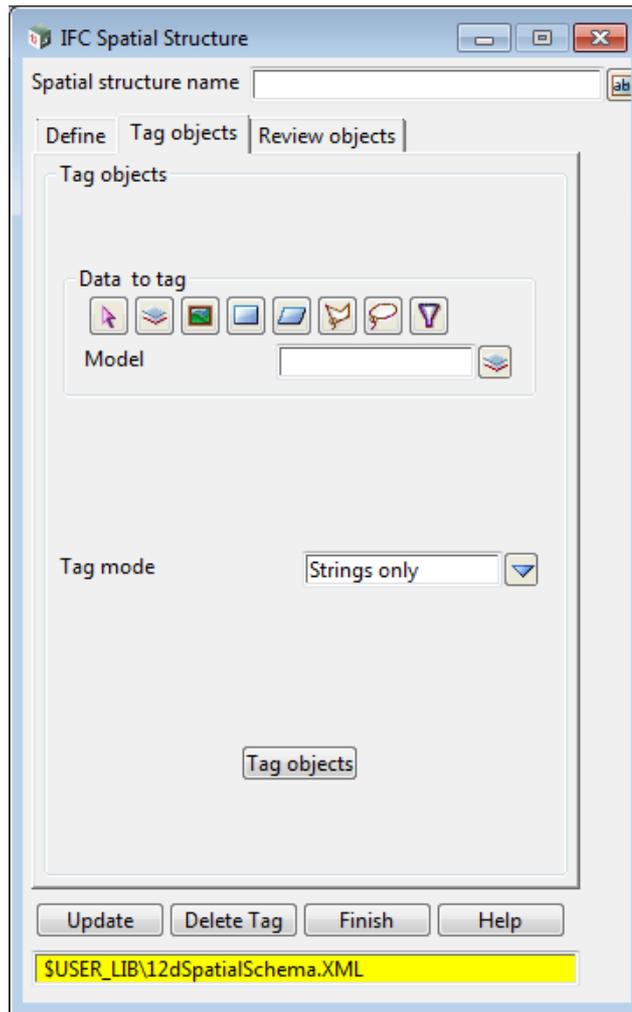
If the **Spatial structure** type is set to anything other than Site, only the Aggregates to and fields that are defined the Spatial Schema File (see [15.2.7.3.1 Spatial Schema File](#)).

Clicking in the **Aggregates to** field brings up a list of Spatial Structures that have already been created. The existing Spatial Structure that this new spatial structure is in is selected. See [11.6.1 Spatial Structures](#).

The values for the other fields can be created/edited and when the **Update** button is pressed, the values are stored as a subgroup of the Tags group of the Project attributes. The subgroup has the name of the Spatial Structure.

### Tag Objects Tab

this tab is used to tag data are being in a given Spatial Structure.



**Data to tag**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data is to be selected for tagging.*

**Tag mode**

choice box

Models only, Strings only, Strings and models

*creates attributes on the selected strings and/or models to indicate which spatial structure the strings and/or models belong to.*

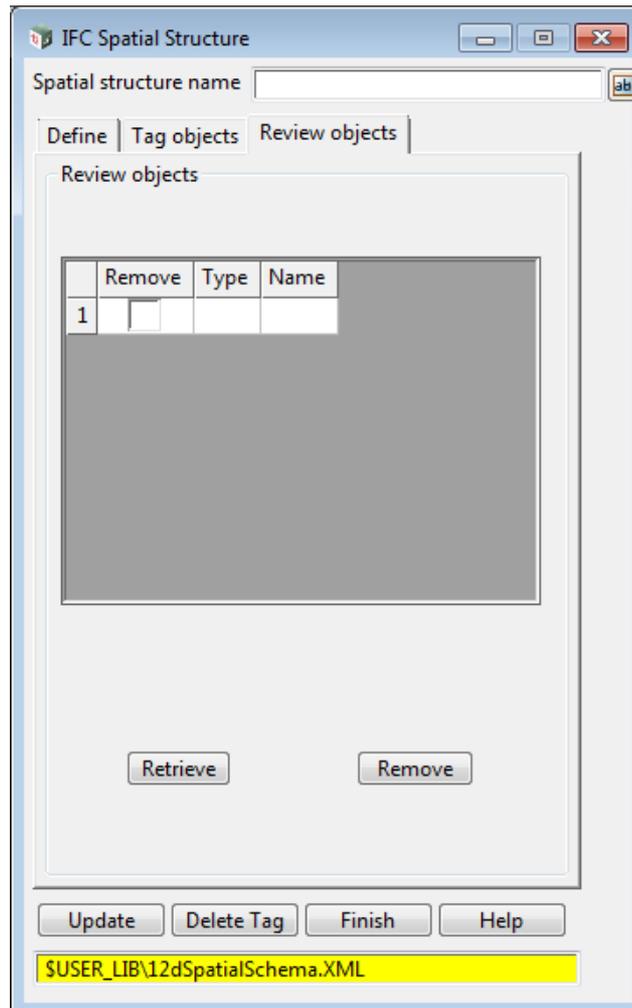
**Tag objects**

button

*tags the selected strings and models with attributes so it is known that they belong to a spatial structure, and which structure it is.*

**Review Objects Tab**

*this tab selects data with a given Spatial Structure and can also remove selected items from the spatial structure.*



**Review Objects Grid**

**Remove** tick box

*if ticked, the string or model in that row will be removed when the **Remove** button is clicked.*

*Whether strings, models, or both are available is dependent on Tag mode selected in Tag objects tab.*

**Type** output String, Model or Trimesh

*the type of the object - string, model or trimesh.*

**Name** output

*the name of the object.*

**Retrieve** button

*adds a selected string or model.*

**Remove** button

*when pressed the data for the given Spatial Container is listed in the grid.*

**Buttons at Bottom**

**Update**

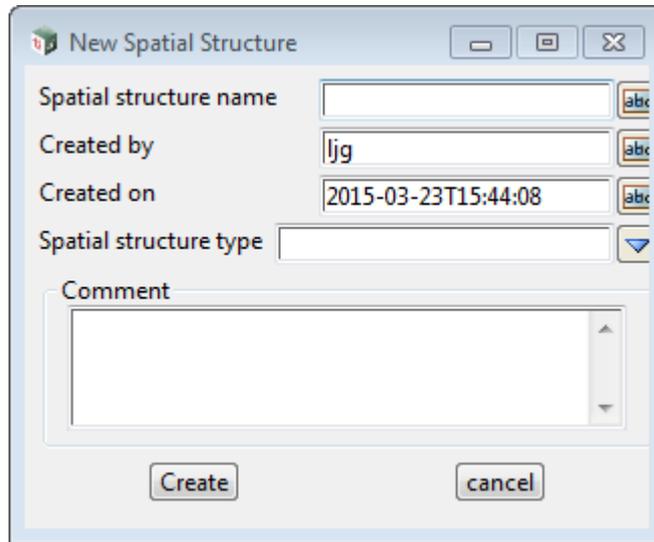
*applies each change. Hit **Update** after every change.*

**Delete Tag**

*deletes the elements with the selected tag.*

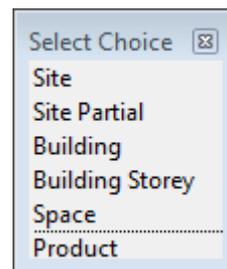
### 15.2.7.4.1 Create Spatial Structure

Selecting New brings up the **Create Spatial Structure** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Container name</b> <i>sets the name of the new container</i>	input		
<b>Created by</b> <i>can be edited to show a different user name if need be</i>	input	user name	
<b>Created on</b> <i>can be edited to show a different date and time</i>	input	current date and time	
<b>Spatial structure type</b>	choice box	as per the spatial schema file.	



*spatial structure types that are defined in the spatial schema file. See [15.2.7.3.1 Spatial Schema File](#)*

<b>Comments</b> <i>allows user comments to be entered.</i>	input	as per New Container panel
<b>Create</b> <i>create the new Spatial Structure.</i>	button	

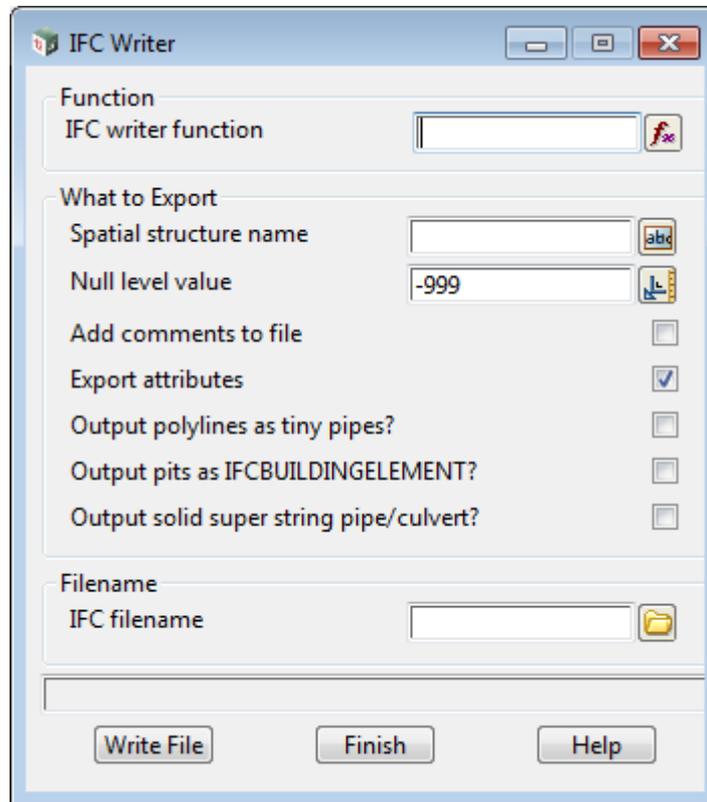
Return to [15.2.7.4 Defining and Assigning Spatial Structures](#).

### 15.2.7.5 IFC Writer

**Position of option on menu:** File I/O =>Data output =>IFC => IFC Writer

The IFC Writer option writes out data with a given tag to a file in the IFC (STEP) format.

Selecting IFC Writer brings up the IFC Writer panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

#### Function

<b>IFC writer function</b>	function box		available IFC functions
----------------------------	--------------	--	-------------------------

*the name for the IFC Writer Function. Note that the IFC Writer is a function so that it can remember the settings in the panel and can be easily rerun as a function at any time.*

#### What to Export

<b>Spatial structure name</b>	defined spatial structures		available spatial structures
-------------------------------	----------------------------	--	------------------------------

*All data with that has been tagged with this IFC spatial structure name, or has been tagged with an IFC spatial structure name that in the Spatial Structure hierarchy under this spatial structure, is written out to the IFC file.*

*So only the name of one spatial structure is required and all the data in that spatial structure and the data in all those spatial structures under it are selected for writing out.*

<b>Add comments to file</b>	tick box	not ticked	
-----------------------------	----------	------------	--

*if **ticked**, comments are added to the IFC file denoting what each group of data is in the IFC file. For example, IFC MODEL INFORMATION, PROJECT INFORMATION and ELEMENTS CONTAINED IN*

*SPATIAL STRUCTURES.*

**Export attributes**                      tick box                      ticked

*if **ticked**, the 12d Project Attributes will be exported to the IFC file to the ifcProject. For 12d Objects that are represented as ifcBuildingElementProxy elements, the string/tin/trimesh attributes are exported as an ifcPropertySet.*

**Export colour table**                      tick box                      not ticked

*if **ticked**, the Colour Table will be exported to the IFC file.*

**Colour Methodology**

**Pset\_Drafting**                      tick box                      ticked (if Export Colour Table is ticked)

*if **Export colour table** is ticked then this is automatically ticked on.*

**Filename**

**IFC Filename**                      file box                      available \*.ifc files

*the name of the IFC file to write all data tagged with the given Spatial Structure.*

**Buttons at bottom**

**Write File**                      button

*write the IFC file.*

Return to [15.2.7 IFC Output](#).

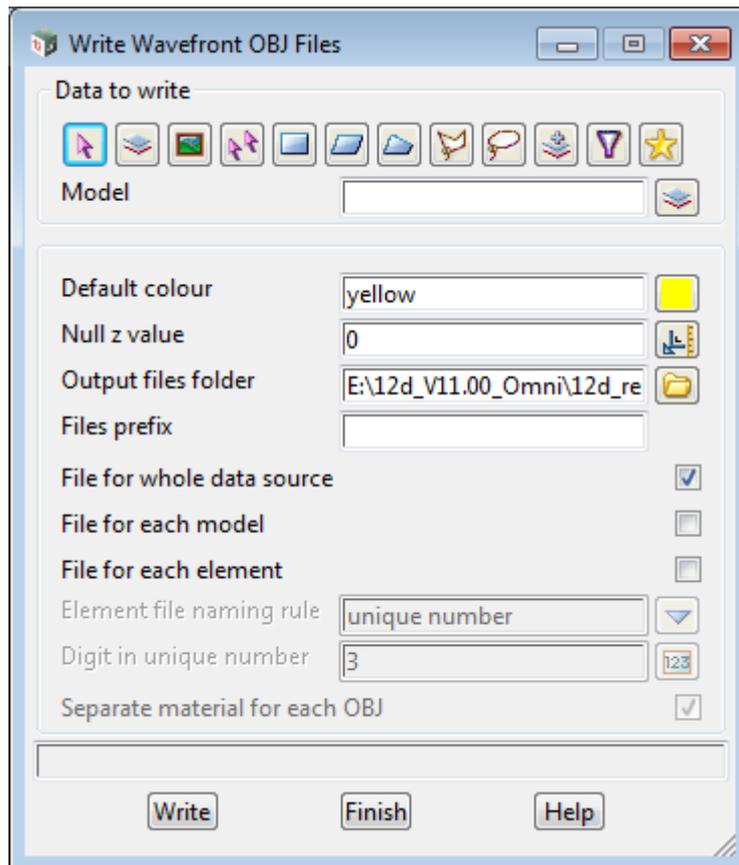
## 15.2.8 Export OBJ

**Position of option on menu:** File I/O =>Data output => OBJ

The OBJ file is a published format from Wavefront for writing out 3d shapes.

Trimeshes, extrusions, pipes and drainage strings can be written to the one OBJ file, or separate OBJ files.

Selecting **OBJ** brings up the **Write Wavefront OBJ Files** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of data to be written out to a STL file.</i>			
<b>Default colour</b>	colour box		
<i>colour for the STL objects.</i>			
<b>Null z value</b>	real box	0	measures menu
<i>value to use in place of any null z values.</i>			
<b>Output files folder</b>	folder box		folder browse
<i>name of the folder to write the obj files to.</i>			

**Files prefix** text box  
*prefix to use in front of any names for the obj files.*

**File for whole data source** tick box  
*if **ticked**, all the 3d shapes in the data source are written out as the one obj. The name of the file is File prefix.*

**File for each model** tick box  
*if **ticked**, an obj file is written out for each model, and all the 3d shapes in the one model are written out as the one obj. The name of the file is the model name prefixed by File prefix.*

**File for each model** tick box  
*if **ticked**, a separate obj file is written out for each model, and all the 3d shapes in the one model are written out as the one obj.*

**File for each element** tick box  
*if **ticked**, a separate obj file is written out for 3d shapes in the data source.*

**Element naming rule** choice box      unique number  
name followed by unique number  
model name followed by unique number  
*if **unique number**, the File prefix followed by a unique number for each shape, is used for the name of each obj file.*  
*If **name followed by unique number**, the File prefix followed by the string name of the 3d shape, followed by a unique number when two string names are the same, is used for the name of each obj file.*  
*If **model name followed by unique number**, the File prefix followed by the name of the model that the 3d shape is in, followed by a unique number for each shape in the model, is used for the name of each obj file.*

**Digits in unique number** integer box  
*number of digits in the unique number. The number will be 0 filled if it has less than this number of digits.*

**Write** button  
*write out the STL file.*

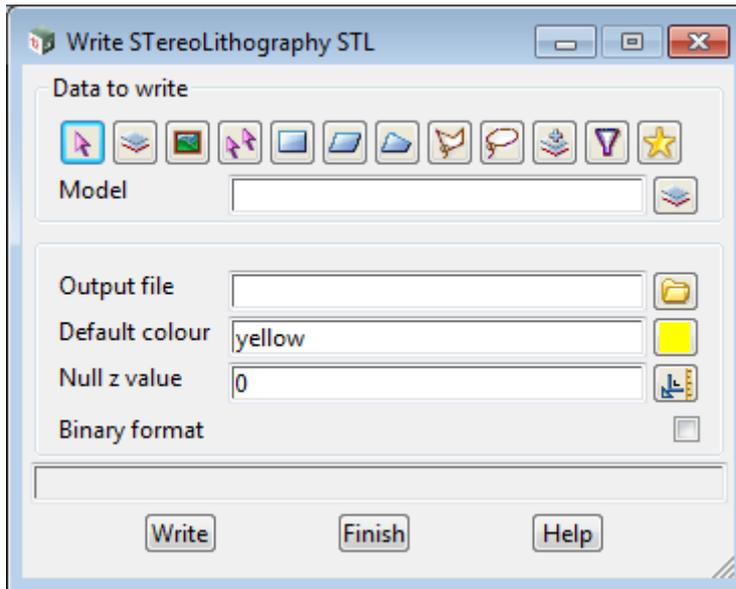
## 15.2.9 Export STL

**Position of option on menu:** File I/O =>Data output =>Export STL

The STL file is a format for driving 3D printers for stereolithography. The STL format only accepts solid objects.

Trimeshes, extrusions, pipes and drainage strings can be written to an STL file.

Selecting **Export STL** brings up the **Write STereoLithography STL** panel:



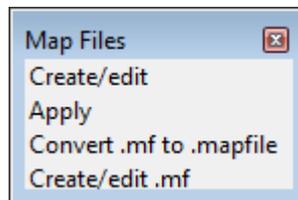
The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b> <i>data selection type.</i>		Model	
<b>Data source</b> <i>source of data to be written out to a STL file.</i>	input		
<b>Output file</b> <i>name of the file for the STL information to be written out to.</i>	folder box		*.stl files
<b>Binary format</b> <i>if <b>ticked</b>, the binary STL format is written out. If <b>not ticked</b>, the ASCII STL format is used.</i>	tick box		
<b>Default colour</b> <i>colour for the STL objects.</i>	colour box		
<b>Write</b> <i>write out the STL file.</i>	button		

## 15.3 Map Files =>Apply

The panel **Map File Apply** for applying a map file was only under **Utilities =>H-Z =>Map** has now also been added as the option

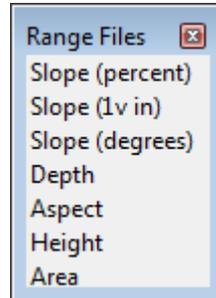
**File =>Map files =>Apply**



# 15.4 Range File Menu

There is a new Area Range file and a new menu option with all the Range files on it.

**File =>Range files**



See

[15.4.1 Area Range File](#)

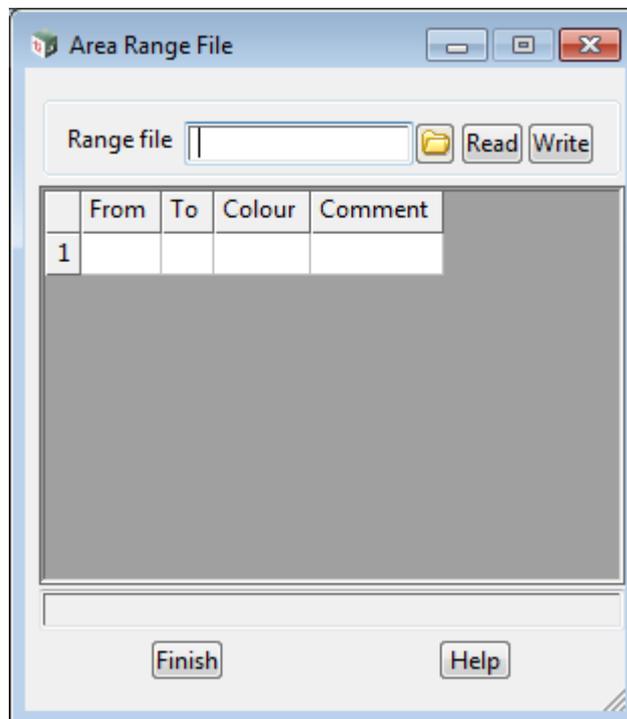
## 15.4.1 Area Range File

For **area colouring**, the range file consists of a list of ranges of areas and colours, one set per line, in the grid

lower\_value upper\_value range\_colour

where this line represents all percentage slopes satisfying

lower\_value < value <= upper\_value.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

- Range file**                      file box                                      available \*.arf files  
*the name of the area range file*
- Read**                                      button  
*read in the file in the Range file field*
- Write**                                      button  
*write out the file in the Range file field*

**Grid Cells**

*for area colouring, the range file consists of a grid of ranges and colours, one set per line, in the grid*

*lower\_value    upper\_value    range\_colour*

*where this line represents all percentage slopes satisfying*

*lower\_value < value <= upper\_value*

**From**                                      real  
*the lower value for this line of the range file*

**To**    real  
*the upper value for this line of the range file*

**Colour**                                      colour box                                      available colours  
*colour for the range*

**Comment**                                      text  
*a comment for this line of the grid*

# 16. Models Menu

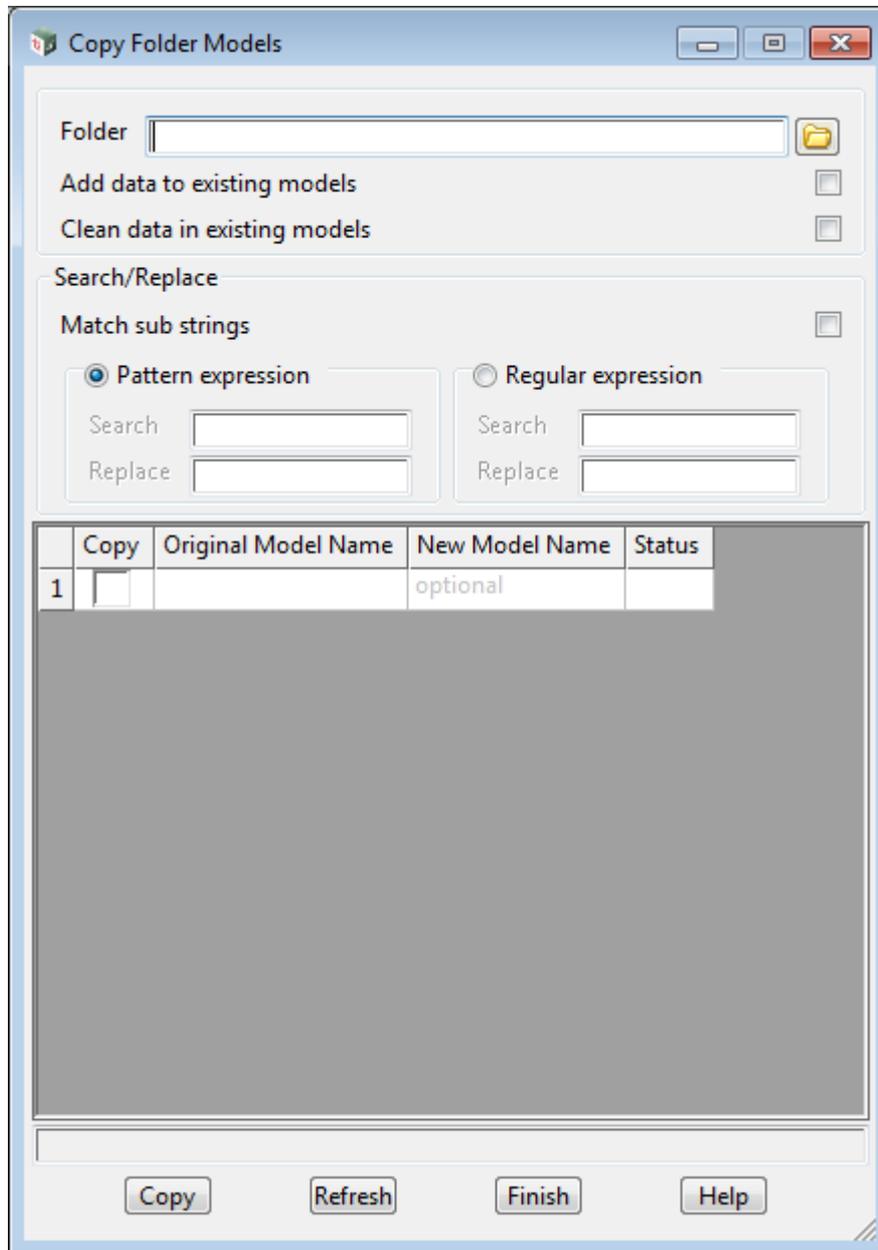
See

[16.1 Copy Folder Models](#)

# 16.1 Copy Folder Models

Position of option on menu: Model =>Utilities =>Copy folder models

Selecting Copy folder models brings up the Copy Folder Models panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Folder</b>	input		Select folder panel

*name of the folder that the \*.model files are in.*

*Once a folder is selected, all the \*.models files in that folder will be listed in the Original Model Name column.*

**Add data to existing models** tick box

*if ticked, if the model that the data is being read into already exists, the new data is copied into the*

model

**Clean data in existing models**    tick box

*if ticked, any existing data in the models being read into, is first cleaned out.*

***NOTE** - If neither tick box is ticked then if the model already exists, **no data is copied**.*

**Search/Replace**

*section for renaming models from the selected folder*

**Match sub strings**                    tick box

*if ticked, the **Search** expression is used to match against part of each model name.*

*If **not ticked**, the **Search** expression is used to match against the entire model name.*

**Pattern expression**                    radio button

*if set on, then **Pattern** expressions given in the **Search** and **Replace** fields are used to modify model names. Pattern expressions include the standard wild card \* and wild character !.*

**Search**                                    input

*pattern to search for in the model names. For example "\* tin" will select all models with a name ending with "tin "*

**Replace**                                    input

*replacement for the search pattern found in the model name. For example, "tin " takes the matched part of the model name and adds " tin " to the front of it.*

*Hence the Search pattern "\* tin" and Replace pattern "tin \*" finds all models with names ending in " tin" and renames them with the name starting with "tin " (and the " tin" at the end of the name is dropped off).*

**Regular expression**                    radio button

*if set on, then **Regular** expressions given in the **Search** and **Replace** fields are used to modify model names.*

**Search**                                    input

*regular expression to search for in the model names.*

**Replace**                                    input

*replacement for the search expression found in the model name.*

**Model Names Grid**

*The models selected by the Search and Replace expressions are shown in the **Original Model Name** column. Note that if the model already exists in the current project, then the cell for that model in **Original Model Name** column will be displayed in yellow.*

**Copy**                                        tick box

*if ticked, the model will be copied.*

*If not ticked, the model will not be copied*

*Clicking RB on **Copy** at the top of the column brings up a menu to **Clear** which turns all the ticks off.*

**Original Model Name**                    column

*name of the existing model in the selected folder.*

**New Model Name**                        column

*if **not blank**, the new name to be given to the copied model.*

*If **blank**, the original model name is used.*

*The **New Model Names** can be from applying the **Search** and **Replace**, or just typing them in.*

**Status**                                      column

*displays if their has been a match or no match for the search/replace for renaming models*

**Copy** button

*copy to this project, the selected models given in the **Original Model Name** field with any new name given in the **New Model Name** field.*

**Refresh** button

*refresh the list of all models in the selected folder.*

# 17. Strings Menu

See

[17.1 Texts Edits](#)

[17.2 Multi String Properties](#)

[17.3 Strings =>Create =>Super](#)

[17.4 Reverse Only Geometry of SA](#)

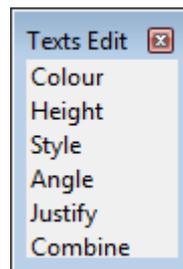
[17.6 Trimesh](#)

[17.7 Super String Fills](#)

[17.8 Strings =>Label =>Cut/fill](#)

## 17.1 Texts Edits

**Strings =>Text edit**



These options are like the **Strings =>Points edit** options in that the options only does the one thing to a selected string. You then pick another string to do the same type of change.

For each of the **Texts Edit** options, press <Esc> or select **Cancel** from the **Pick Ops** menu to terminate the option. Selecting another CAD option will also terminate the option and start the new option.

See

[17.1.1 Change Text Colour](#)

[17.1.2 Change Text Height](#)

[17.1.3 Change Text Style](#)

[17.1.4 Change Text Angle](#)

[17.1.5 Change Text Justify](#)

[17.1.6 Change Combine](#)

## 17.1.1 Change Text Colour

Click on **Colour** and a *Colour Typed Input* box comes up



The new colour is picked from the pop up list, or typed into the box and <Enter> pressed. The box is then removed and the following message is written to the screen message area

```
<Pick text to change (c)olour [orange]> [picks][fast][Menu]
```

Pick text and its colour will be changed to that shown in the square brackets ([]). Then pick another text to change its colour.

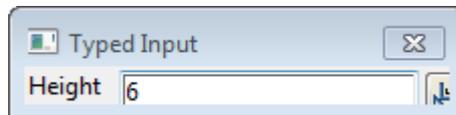
Typing **c** or **C** brings up the *Colour Typed Input* box to select a new colour.

All subsequent selected texts will have their colour changed to this colour until either the option is terminated or a new colour is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

## 17.1.2 Change Text Height

Click on **Height** and a *Height Typed Input* box comes up



The new height is typed into the box and <Enter> pressed.

The box is then removed and the following message is written to the screen message area

```
<Pick text to change (h)eight [6.00000]> [picks][fast][Menu]
```

Pick text and its height will be changed to that shown in the square brackets ([]). Then pick another text to change its height.

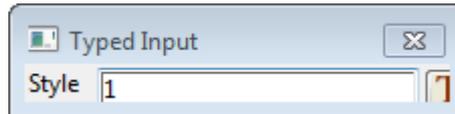
Typing **h** or **H** bring up the *Height Typed Input* box to select a new height.

All subsequent selected texts will have their height changed to this height until either the option is terminated or a new height is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

## 17.1.3 Change Text Style

Click on **Style** and a *Style Typed Input* box comes up



The new style is picked from the pop up list, or typed into the box and <Enter> pressed.

The box is then removed and the following message is written to the screen message area

```
<Pick text to change (s)tyle [1]> [picks][fast][Menu]
```

Pick text and its style will be changed to that shown in the square brackets ([]). Then pick another text to change its style.

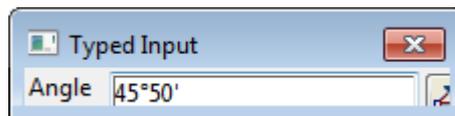
Typing **s** or **S** brings up the **Style Typed Input** box to select a new style.

All subsequent selected texts will have their style changed to this style until either the option is terminated or a new style is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

## 17.1.4 Change Text Angle

Click on **Angle** and an **Angle Typed Input** box comes up



The new angle is typed into the box and <Enter> pressed.

The box is then removed and the following message is written to the screen message area

```
<Pick text to change (a)ngle [ 45° 50' 0"]> [picks][accepts][Menu] "T
```

Pick text and its angle will be changed to that shown in the square brackets ([]). Then pick another text to change its angle.

Typing **a** or **A** brings up the **Angle Typed Input** box to select a new angle.

All subsequent selected texts will have their angle changed to this angle until either the option is terminated or a new angle is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

### Note

The angle is type in in **hp notation** and measured counter clockwise from the positive x-axis. The box will convert the hp notation and display it in degrees, minutes and seconds. For example, 45.5 in hp notation become 45 degrees 50 minutes.

## 17.1.5 Change Text Justify

Click on **Justify** and a **Justify Typed Input** box comes up



The new justification is picked from the pop up list, or typed into the box and <Enter> pressed. The box is then removed and the following message is written to the screen message area

<Pick text to change (j)ustify [bottom-left]> [picks][fast][Menu]

Pick text and its justification will be changed to that shown in the square brackets ([]). Then pick another text to change its justification.

Typing **j** or **J** brings up the **Justify Typed Input** box to select a new justification.

All subsequent selected texts will have their justification changed to this justification until either the option is terminated or a new justification is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

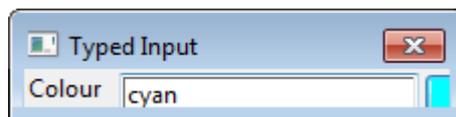
## 17.1.6 Change Combine

This option allows you to change one or more of Colour, Height, Angle, Textstyle and Justification.

Selecting **Combine** writes the following message to the screen message area

<Pick text to change (c)olour (h)eight (a)ngle (s)tyle (j)ustify> [picks][fast][Menu]

Type **c** or **C** and a **Colour Typed Input** box comes up



The new colour is picked from the pop up list, or typed into the box and <Enter> pressed.

The box is then removed and the following message is written to the screen message area

<Pick text to change (c)olour[cyan] (h)eight (a)ngle (s)tyle (j)ustify> [picks][fast][Men]

This **sets** the colour to cyan.

Pick text and its colour will be changed to that shown in the square brackets ([]). Then pick another text to change its colour.

Typing **c** or **C** when there is a colour shown between the square brackets (for example [cyan]), **removes** the colour from the square brackets.

If any text is picked when there is no colour in the square brackets then its colour will **not** be changed.

Similarly:

### Height

**IF** there is no height in square brackets after (h)eight, typing **h** or **H** brings up the **Height**

**Typed Input** box to select a new height. Entering a height and pressing <Enter> **sets** the new height and it will appear in square brackets after *(h)eight* in the message written to the screen message area.

Typing **h** or **H** when there is a height shown between the square brackets (for example [2.00]), **removes** the height from the square brackets.

**Angle**

**IF** there is no angle in square brackets after *(a)ngle*, typing **a** or **A** brings up the *Angle Typed Input* box to select a new angle. Entering an angle and pressing <Enter> **sets** the new angle and it will appear in square brackets after *(a)ngle* in the message written to the screen message area.

Typing **a** or **A** when there is an angle shown between the square brackets, **removes** the angle from the square brackets.

**Textstyle**

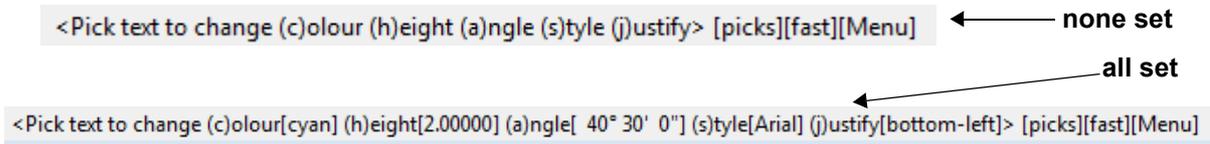
**IF** there is no textstyle in square brackets after *(s)tyle*, typing **s** or **S** brings up the *Style Typed Input* box to select a new textstyle. Entering a textstyle and pressing <Enter> **sets** the new textstyle and it will appear in square brackets after *(s)tyle* in the message written to the screen message area.

Typing **s** or **S** when there is a textstyle shown between the square brackets, **removes** the textstyle from the square brackets.

**Justification**

**IF** there is no justification in square brackets after *(j)ustify*, typing **j** or **J** brings up the *Justify Typed Input* box to select a new justification. Entering a justification and pressing <Enter> **sets** the new justification and it will appear in square brackets after *(j)ustify* in the message written to the screen message area.

Typing **j** or **J** when there is a justification shown between the square brackets, **removes** the justification from the square brackets.



If any text is picked then the values of the text are changed for any that there is a value in the square brackets otherwise that value of the text is not changed.

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

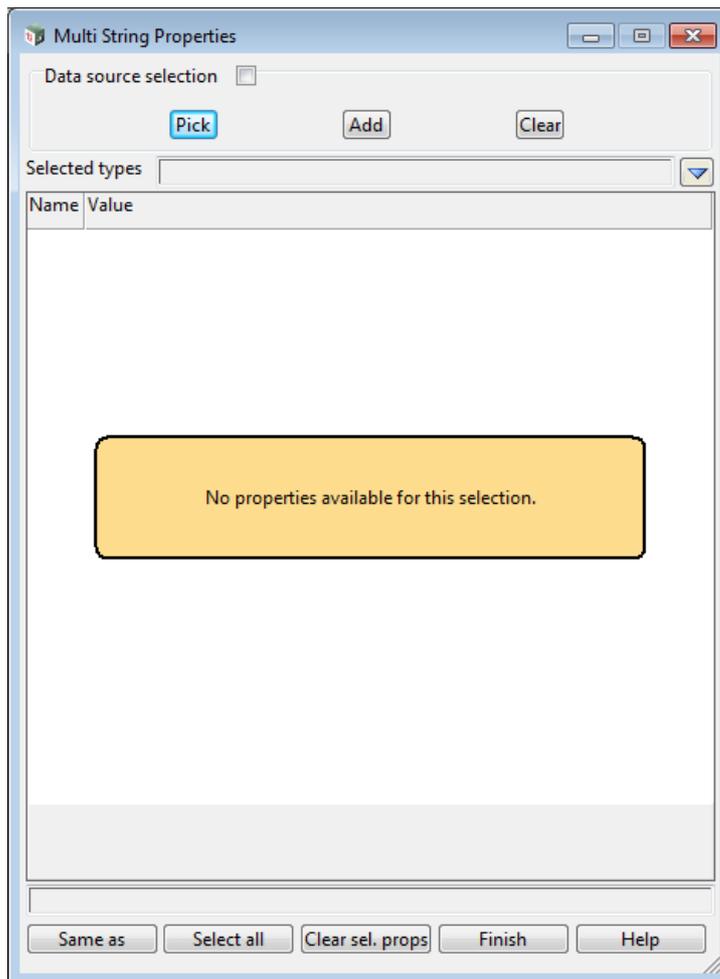
## 17.2 Multi String Properties

**Position of option on menu:** Strings =>Properties =>Multi string

The **Multi String Properties** panel allows you to select strings and the panel displays all the common properties of the selected strings, and the value of a property if it is the same for all the selected elements.

The value of any of these properties can then be modified and all the selected strings will have the property changed to the modified value.

Selecting **Multi strings** brings up the **Multi String Properties** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source selection</b>	tick box	not ticked	

*if ticked, a Data Source is displayed and used to define the strings in the Selection Set.*

**Data Source** data source box

*all the strings in the Data Source are selected for the Selection Set.*

*If not ticked, **Add** and **Clear** buttons are displayed and used to defined the strings in the selection Set.*

**Add** button

when **Add** is clicked, strings are selected/deselected using the *Multipick* method, and are added to any string already selected for the Selection Set by the **Add** button.

The **Add** button can be used any number of times and each time it adds the new selected items to the Selection Set.

**Clear** button

When **Clear** is clicked, the Selection Set is cleared. That is, there are no strings in the Selection Set.

The common properties of all the strings in the Selection Set are displayed in the **Name-Value** property list. If the value of a common property is the same for **all** the strings in the Selection Set, then the common value is displayed in the **Value** column for that property. Otherwise the **Value** column for that property is left blank.

**Common Property List**

the Common Property Tree displays all the common properties of the selected strings that can be modified.

The name of each common property is displayed in a row in the **Name** column.

For a particular common property, if all the selected strings have the same value for that common property then the common value is displayed in the **Value** column for the row of the common property. Otherwise the value column is blank.

If a value is **modified** in the **Value** column, then for all the strings in the Selection Set, that property is **changed** to the new value.

**Same As, Select All and Clear Selection Buttons**

these three buttons work together. Rows in the **Property List** can be selected (highlighted), and then the **Same As** button used to select a string whose properties are used to update the strings in the selection set for the highlighted common properties.

**Same As** button

when **Same As** is clicked and a string is selected, then for all the strings in the Selection Set, the values of the highlighted properties in the **Common Property List** are set to the values of the **Same As** string.

**Select All** button

clicking **Select All** highlights all the rows of properties in the **Property List**. If required, individual rows can then be deselected using *Ctrl-LB*.

**Clear Property Selection** button

clicking **Clear Property Selection** clears all the highlighted property rows.

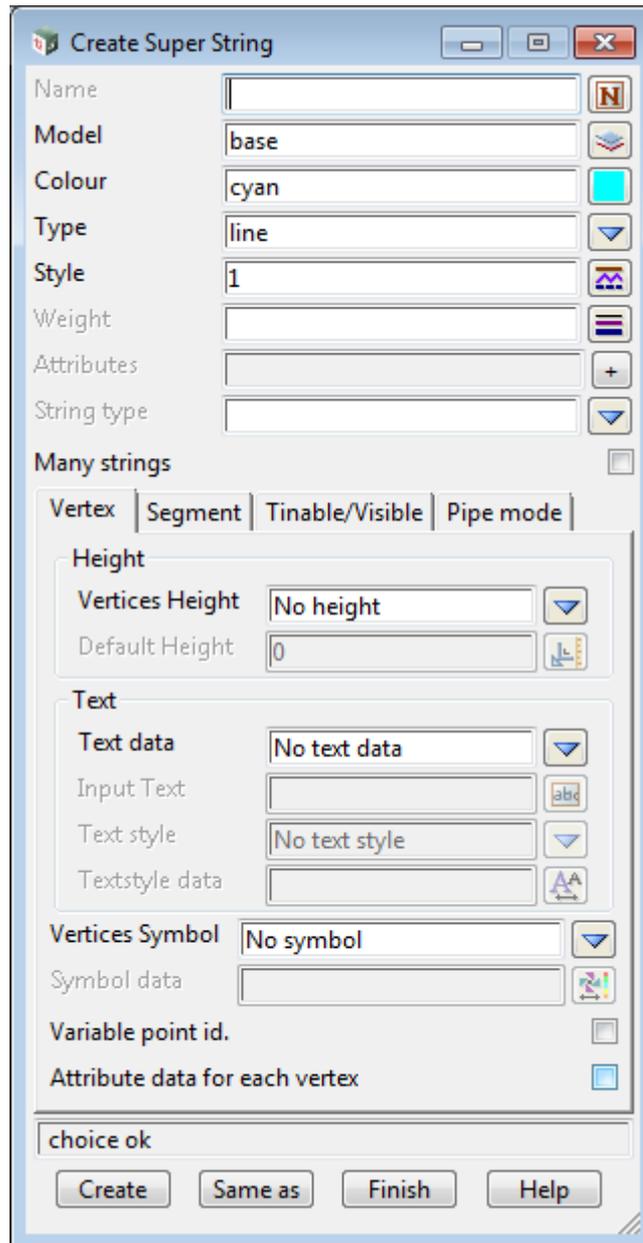
## 17.3 Strings =>Create =>Super

For V11, the Create Super String panel has been rearranged and also takes values from the CAD ControlBar, the Attributes ControlBar and the Pipe ControlBar.

If the **Name** icon is used in the **Create Super String** panel to select a name from the names.4d pop up, then the panel fields are updated from the names.4d file, including the new **Attribute Data** and **Pipe** sections of the names.4d file.

**Position of option on menu:** Strings =>Create =>Super

Selecting Super string displays the **Create Super String** panel.



When the panel opens, various panel field values are taken from the **ControlBars**.

The information in **Name**, **Model**, **Colour**, **Style** and **Tunable** are taken from the CAD ControlBar, **Attributes** from the Attributes ControlBar and **Pipe** information from the Pipe ControlBar.

If the **Name** icon is used in the **Create Super String** panel to select a name from the names.4d pop up, then the panel fields are updated from the names.4d file.

To **create** a new super string, the appropriate panel fields are filled in and the **Create** button clicked.

The new fields and buttons used in the **Create Super String** panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Name</b> <i>the name of the new string.</i>	names.4d box		names.4d file
<b>Model</b> <i>name of the model for the new string.</i>	model box		available models
<b>Colour</b> <i>the colour of the new string.</i> <i>Note that each segment can have a different colour if in the <b>Segment</b> tab, the <b>Segment colour</b> choice is <b>Varied</b>.</i>	colour box	default colour	available colours
<b>Type</b> <i>breakline type (point-line type) of the string</i> <i>As an overall string property:</i> <i>If a string has type <b>line</b>, then all vertices and all segments are tinnable.</i> <i>If a string has type <b>point</b>, then all vertices are tinnable but the segments are not tinnable.</i>  <i>Note that each vertex and segment can have different tinnabilities and it is controlled in the <b>Tinnable</b> tab.</i>	choice box	line	line, point
<b>Style</b> <i>line style of the new string.</i>	linestyle box	1	available line styles
<b>Weight</b> <i>thickness of the string when plotted.</i>	weight box	0	
<b>String type</b> <i>if <b>not blank</b>, the like type sets a number of the panel field settings.</i>	choice box		2d like, 3d like, 4d like
<b>Many strings</b> <i>if ticked then after the current string is finished, a new <b>Create Super String</b> panel is placed on the screen with all the same values for the panel fields as in the current panel.</i>	tick box		

### Vertex tab

*The fields for setting up what information is required for vertices.*

<b>Vertices height</b>	choice box	Constant for all Varied No height
<i>if <b>Constant for all</b>, there is only one height that is used for all vertices of the string (2d like).</i>		
<i>If <b>Varied</b>, there is a different height for each vertex (3d like)</i>		
<i>If <b>No height</b>, there is no height at all for the string.</i>		

*if the **Vertices height** choice is **Constant for all** then the field **Default height** is enabled.*

<b>Default height</b>	real box
<i>the one height that is used for every vertex of the string.</i>	

<b>Text data</b>	choice box	Constant for all
------------------	------------	------------------

Varied  
No text data

*if Constant for all, there is one text for all the vertices of the string.  
If Varied, there is different text for each vertex.  
If No text data, there is no text at all for the string.*

*if the Text data choice is Constant for all then the fields Input text and Text style are enabled.*

**Input text**                      text box

*the one bit of text that is used for every vertex of the string.*

**Text style**                      choice box

Constant for all  
Varied  
No text style

*if Constant for all, there is one textstyle that is used for all the vertices of the string.  
If Varied, there is a different textstyle for each vertex.  
If No text data, there is no textstyle at all for the string.*

*if the Text style choice is Constant for all then the field Textstyle data is enabled.*

**Textstyle data**                      textstyle data box

*the one textstyle data that is used for every vertex of the string.*

**Symbol data**                      choice box

Constant for all  
Varied  
No symbol

*if Constant for all, there is one symbol that is used for all vertices of the string.  
If Varied, there is a different symbol for each vertex  
If No symbol, there is no symbols at all for the string.*

*if the Symbol data choice is Constant for all then the field Symbol data is enabled.*

**Symbol data**                      symbol data box

*the one symbol data that is used for every vertex of the string.*

**Variable point id**                      tick box

*if ticked, there is a different point id for each vertex.  
If not ticked, there are no vertex ids for any vertex of the string.*

**Attribute data for each vertex**                      tick box

*if ticked, there can be attributes for each vertex.  
If not ticked, there are no attributes on any vertex.*

## Segment tab

*The fields for setting up what information is required for segments.*

**Segments colour**                      choice box

Constant for all  
Varied

*if Constant for all, there is one colour for every segment of the string and that is the colour in the Colour field.*

*If Varied, there is a different colour for each segment.*

**Text data**                      choice box

Constant for all  
Varied

No text data

*if **Constant for all**, there is one text for all the segments of the string.  
If **Varied**, there is different text for each segment.  
If **No text data**, there is no text at all for the string.*

*if the **Text data** choice is **Constant for all** then the fields **Input text** and **Text style** are enabled.*

**Input text** text box

*the one bit of text that is used for every segment of the string.*

**Text style** choice box Constant for all  
Varied  
No text style

*if **Constant for all**, there is one textstyle that is used for all the segments of the string.  
If **Varied**, there is a different textstyle for each segment.  
If **No text data**, there is no textstyle at all for the string.*

*if the **Text style** choice is **Constant for all** then the field **Textstyle data** is enabled.*

**Textstyle data** textstyle data box

*the one textstyle data that is used for every segment of the string.*

**Radius/Major** tick box

*if ticked, each segment can have a radius and a major flag.  
If not ticked, all segments are straight lines.*

***Note:** the radius is a signed value and in the direction of travel along the arc, a negative radius means the arc bends to the left and a positive radius means the arc bends to the right. However this is still not enough to uniquely define the arc and it can be a minor major arc (turns through less than or equal to 180 degrees), or a major arc (turns through more than 180 degrees). The Major flag is 0 for a minor arc and 1 for a major arc.*

**Attribute data for each segment** tick box

*if ticked, there can be attributes for each segment.  
If not ticked, there are no attributes on any segment.*

## Tinable/Visible tab

*The fields for setting up what tinability is allowed for vertices and segments.*

***Tinability** for a **vertex** is the property that the vertex is included in a tin (triangulation).*

*If a vertex is tinable then it is included in a tin.*

*If a vertex is not tinable, then it is not included in a tin.*

***Tinability** for a **segment** is the property that the segment is to be included in a tin (triangulation). That is, the segment is to be a side of a triangle in a tin. Another name for a tinable segment is that it is a **breakline**. Even though a segment is tinable, it may not end up in a tin if there are crossing breaklines. Also a tinable segment can not be included in a tin if either of its end vertices are not tinable.*

*If a segment is tinable then if possible, it is included as the side of a triangle in a tin.*

*If a segment is not tinable, then it does not have to be included as a side of a triangle in a tin.*

*As an overall string property,:*

*If a string has type **line**, then all vertices and all segments are tinable.*

*If a string has type **point**, then all vertices are tinable but the segments are not tinable.*

**Vertex tinability** choice box Constant

Varied  
Not tinable

*if **Constant**, there is one tinability and it is the same for all vertices in the string.  
If **Varied**, each vertex can have a different tinability.  
If **Not tinable**, then all vertices are not tinable.*

**Segment tinability**                      choice box                      Constant  
Varied  
Not tinable

*if **Constant**, there is one tinability and it is the same for all segments in the string.  
If **Varied**, each segment can have a different tinability.  
If **Not tinable**, then all segments are not tinable.*

**Pipe mode tab**

*The fields for setting up what pipe modes are allowed for segments.  
Segments can be all round for a string, or all box (culvert) for a string but can't be a mixture of round and box in the one string.  
If a segment is round then it just needs a diameter.  
If a segment is culvert (or box), it needs a width and a height.*

**Pipe choice**

*There are five choice: the string is a  
constant pipe string and has one diameter for the entire string  
a variable pipe string and can have a different diameter for each segment  
a constant culvert string and has the one width and height for each segment  
a variable culvert string and can have a different width and height for each segment  
or the string has no diameter or width and height for any segment.*

**Pipe mode**                      choice box                      no round pipe or culvert  
round pipe entire string  
round pipe each segment  
culvert entire string  
culvert each segment

*if **no round pipe or culvert**, there is no diameters, or any widths or heights for any segment.  
if **round pipe entire string**, there is **one** diameter for all segments in the string.  
if **round pipe each segment**, there is a different diameter for each segments of the string.  
if **culvert entire string**, there is **one** width and a height for all segments in the string.  
if **culvert each segment**, there is a different width and height for each segment of the string.*

*if the Pipe mode choice is **round pipe entire string** then the field **Diameter** is enabled.*

**Diameter**                      real box  
*diameter to use for every segment of the string.*

*if the Pipe mode choice is **culvert entire string** then the fields **Width** and **Height** are enabled.*

**Width/Height**                      real box  
*width/height to use for every segment of the string.*

*if the Pipe mode choice is **NOT no round pipe or culvert** then the field **Justify** is enabled.*

**Justify**                      choice box                      Invert, centre, obvert  
*If **invert**, the coordinates of the vertices are on the top of the pipe/culvert.  
If **centre**, the coordinates of the vertices are in the centre of the pipe/culvert.  
If **obvert**, the coordinates of the vertices are at the bottom of the pipe/culvert.*

## Buttons at bottom

**Create** button

*After the **create** button is chosen, the **super edit** menu and **super edit info** panel are displayed.*

**Same as** button

*After the **same as** button is chosen, another string is selected and information about it is used for the fields in this panel.*

## 17.4 Reverse Only Geometry of SA

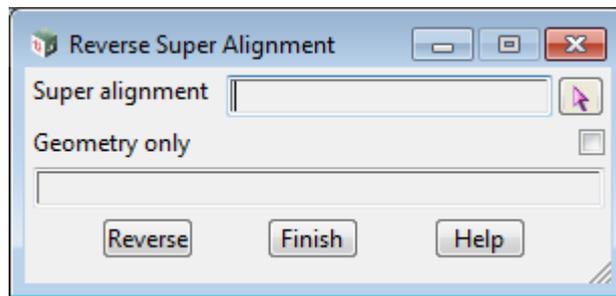
**Position of option on menu:** Strings => Super Alignments => Tools => Reverse

On the **Reverse Super Alignment** panel, there is now a **Geometry only** tick box.

If it is ticked **on** then only the Geometry (the segments for the horizontal and the segments for the vertical) from the Super Alignment is used (which consists of straights, arcs, parabolas, transitions and offset transitions) and a reverse string is created by simply reversing the segments.

The reversed string no longer has any constructive definition but just consists of the horizontal and vertical data.

This is useful when strings come in from other systems and have geometry (that is horizontal and vertical segments) but won't solve. These strings can now be reversed.



**Geometry only**                      tick box                      not ticked

*if **ticked** then only the segments of the horizontal and vertical geometry are reversed. All construction data is removed. So this will work for super alignments that have geometry but won't solve.*

*if **not ticked** then the string will need to solve before it can be reversed.*

**Note:** this reverses and replaces the existing string so if you don't want to lose the data in your existing string, make a copy of it first.

## 17.5 Offset Mode in Parallel SA

**Position of option on menu:** Strings =>Super Alignments =>Tools =>Parallel

On the **Parallel Super Alignment** panel, there is now an **Offset mode**.

**Offset mode** choice box fix and free, floating, ip

*if **fix and free**, fixed free fixed elements will be used to parallel the super alignment.*

*If **floating**, the super alignment will start with a fixed element followed by floating elements.*

*If **ip**, an IP based super alignment will be created.*

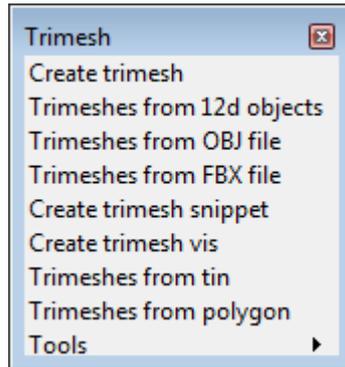
## 17.6 Trimesh

There is a new menu

**Strings =>Trimesh**

The options on this menu create trimeshes and snippets to create trimeshes when used in an Apply MTF, creates trimeshes when reading in files and also trimeshes with vertical thickness from a polygon or tin.

**Current position on menu: Strings =>Trimesh**



See

[17.6.1 Trimeshes from 12d Objects](#)

[17.6.2 Trimeshes from Wavefront OBJ File](#)

[17.6.2 Trimeshes from Wavefront OBJ File](#)

[17.6.3 Trimeshes from FBX File](#)

[17.6.4 Trimeshes from Tin](#)

[17.6.5 Trimeshes from Polygon](#)

[17.6.6 Trimesh Tools](#)

## 17.6.1 Trimeshes from 12d Objects

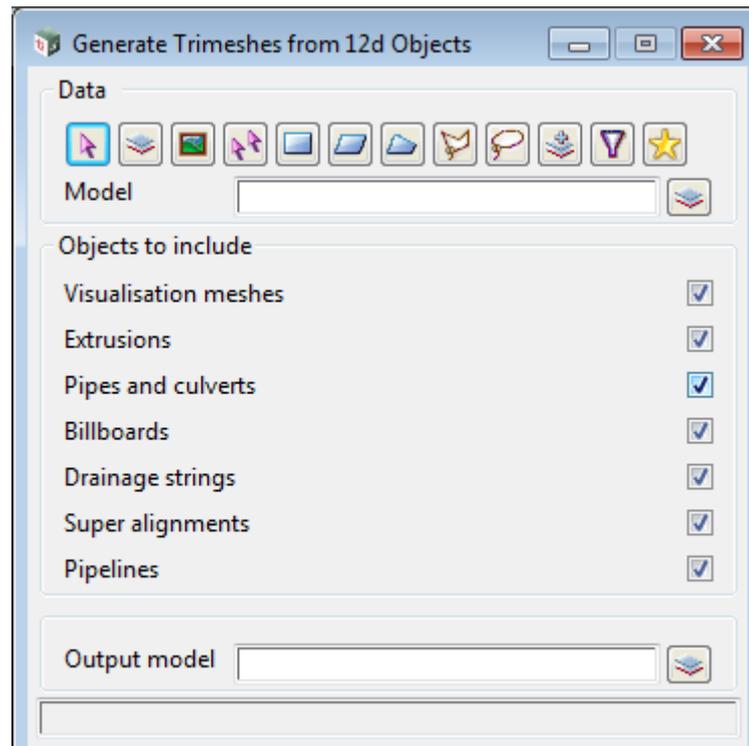
### Generate\_Trimeshes\_from\_12d\_Objects

Position of option on menu: **Strings =>Trimesh =>Trimeshes from 12d objects**

This option creates trimeshes for any mesh, extrude, super string pipe or culvert, drainage/sewer string or billboard, in the selected data.

The trimeshes can then be added to a section view, or cross section plots, to see that they are correctly sectioned through.

Selecting **Trimeshes from 12d Objects** brings up the **Generate Trimeshes from 12d Objects** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to process. For any meshes, a trimesh is created.*

**Visualisation meshes**

tick box

tick

*if **ticked**, any meshes in the data source are converted to trimeshes.  
If **not ticked**, then meshes are not converted to trimeshes.*

**Extrusions**

tick box

tick

*if **ticked**, any extrudes in the data source are converted to trimeshes.  
If **not ticked**, then extrudes are not converted to trimeshes.*

**Pipes and culverts**

tick box

tick

*if **ticked**, any super string pipe or culvert in the data source are converted to trimeshes.  
If **not ticked**, then super string pipe or culverts are not converted to trimeshes.*

- Billboard**                      tick box                      tick  
*if **ticked**, any billboard in the data source are converted to trimeshes.  
If **not ticked**, then billboards are not converted to trimeshes.*
- Drainage strings**                      tick box                      tick  
*if **ticked**, any drainage/sewer strings in the data source are converted to trimeshes.  
If **not ticked**, then drainage/sewer strings are not converted to trimeshes.*
- Output model**                      model box                      available models  
*model to add all the created trimeshes to.*
- Generate**                      button  
*create trimeshes for all the selected type that are in the data source.*
- Continue to [17.6.2 Trimeshes from Wavefront OBJ File](#) or return to [17.6 Trimesh](#).

## 17.6.2 Trimeshes from Wavefront OBJ File

**Position of option on menu:** File I/O =>Data input =>OBJ

**Position of option on menu:** Strings =>Trimesh =>Trimeshes from OBJ file

This is documented in [9.1.18 Wavefront OBJ Input](#).

Continue to [17.6.3 Trimeshes from FBX File](#) or return to [17.6 Trimesh](#).

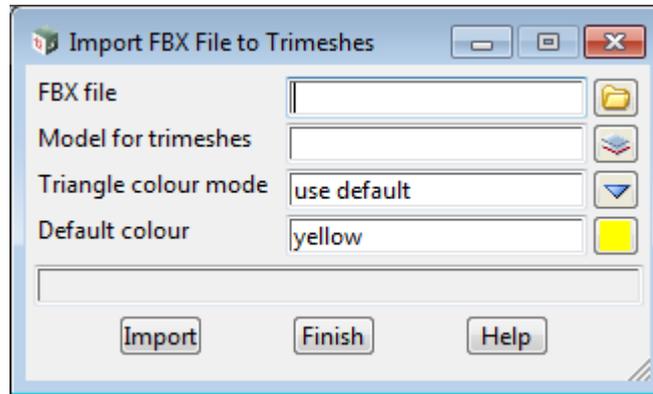
## 17.6.3 Trimeshes from FBX File

**Position of option on menu:** Strings =>Trimesh =>Trimeshes from FBX file

**Position of option on menu:** File I/O =>Data input =>FBX

FBX (Filmbox) is a proprietary format developed by Kaydara and owned by AutoDesk since 2006, to provide interoperability between digital content creation applications.

This option reads in a FBX file and converts all possible objects to trimeshes.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>FBX file</b> <i>name of the FBX file to be read in</i>	file box		*.fbx files
<b>Model for trimeshes</b> <i>model to add all the created trimeshes to.</i>	model box		available models
<b>Triangle colour mode</b> <i>if user default, use the Default colour for the trimeshes. If RGB diffuse colour, use the RGB diffuse colour of the FBX triangle object as the triangle colour. If material name, use the material name of the FBX triangle as the triangle colour.</i>	choice box	use default	use default, RGB of diffuse colour material name
<b>Default colour</b> <i>default colour for the trimeshes.</i>	colour box	yellow	available colours
<b>Import</b> <i>read the FBX file and where ever possible create trimeshes from the FBX data.</i>	button		

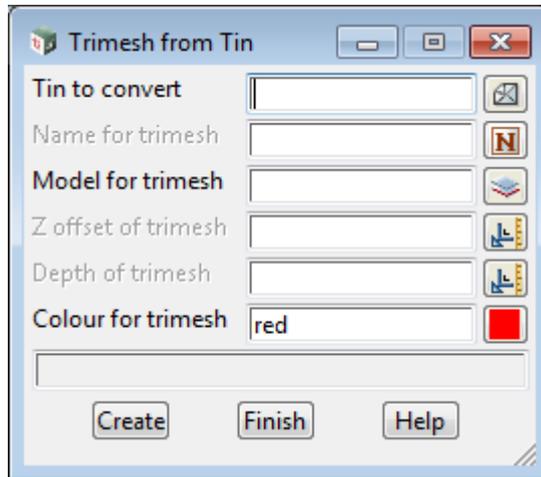
Continue to [17.6.4 Trimeshes from Tin](#) or return to [17.6 Trimesh](#).

## 17.6.4 Trimeshes from Tin

Position of option on menu: **Strings =>Trimesh =>Trimeshes from tin**

This option takes a tin and creates a trimesh by first moving the tin up vertically by a user given Z offset of trimesh to form the top of the trimesh, and then copying the top a vertical down by a user given Depth of trimesh.

Selecting **Trimeshes from tin** brings up the **Generate Trimesh from tin** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tin to convert</b> <i>tin to create a trimesh from.</i>	tin box		
<b>Name for trimesh</b> <i>name for the created trimesh.</i>	name box		
<b>Model for trimesh</b> <i>model to add the created trimesh to.</i>	model box		available models
<b>Z offset of trimesh</b> <i>the top of the trimesh is the tin translated vertically up by this value. This value can be negative and then the top of the trimesh is lower rather than above the tin.</i>	real box		
<b>Depth Z offset of trimesh</b> <i>the bottom of the trimesh is this value vertically below the top of the trimesh. This value can be negative then the "bottom" of the trimesh is above the "top" of the trimesh.</i>	real box		
<b>Colour for trimesh</b> <i>colour for the created trimesh.</i>	colour box		available colours
<b>Create</b> <i>create trimesh from the tin.</i>	button		

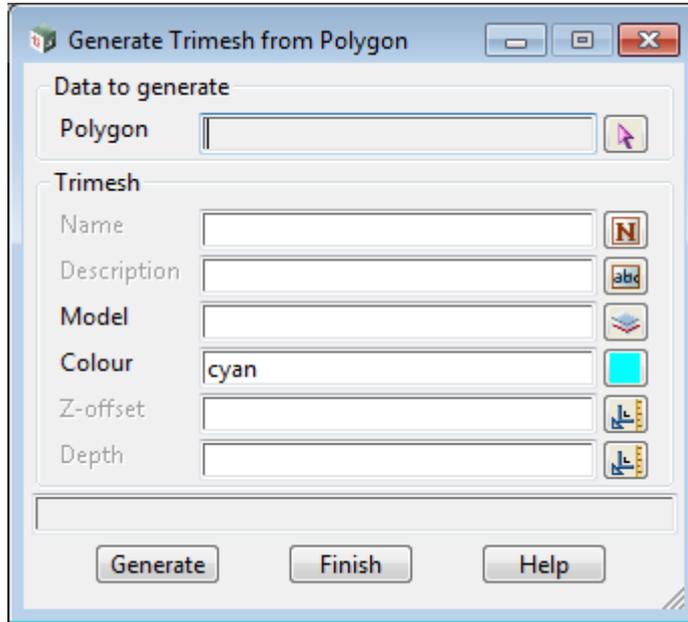
Continue to [17.6.5 Trimeshes from Polygon](#) or return to [17.6 Trimesh](#) .

## 17.6.5 Trimeshes from Polygon

Position of option on menu: **Strings =>Trimesh =>Trimeshes from polygon**

This option takes a polygon and first creates a tin of the polygon, and then creates a trimesh by first moving the tin up vertically by a user given Z offset of trimesh to form the top of the trimesh, and then copying the top a vertical down by a user given Depth of trimesh.

Selecting **Trimeshes from polygon** brings up the **Generate Trimesh from Polygon** panel.



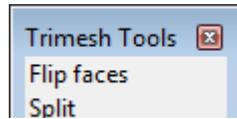
The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Polygon</b> <i>polygon to create the trimesh from.</i>	string select		
<b>Name</b> <i>name for the created trimesh.</i>	name box		
<b>Description</b> <i>a description that is recorded in the trimesh.</i>	input		
<b>Model</b> <i>model to add the created trimesh to.</i>	model box		available models
<b>Colour</b> <i>colour for the created trimesh.</i>	colour box		available colours
<b>Z-offset</b> <i>the top of the trimesh is the tin of the polygon translated vertically up by this value. This value can be negative and then the top of the trimesh is lower rather than above the tin of the polygon.</i>	real box		
<b>Depth</b> <i>the bottom of the trimesh is this value vertically below the top of the trimesh of the polygon. This value can be negative then the "bottom" of the trimesh is above the "top" of the trimesh.</i>	real box		
<b>Generate</b> <i>create trimesh from the polygon.</i>	button		

Continue to [17.6.6 Trimesh Tools](#) or return to [17.6 Trimesh](#).

## 17.6.6 Trimesh Tools

Position of option on menu: **Strings =>Trimesh**



See

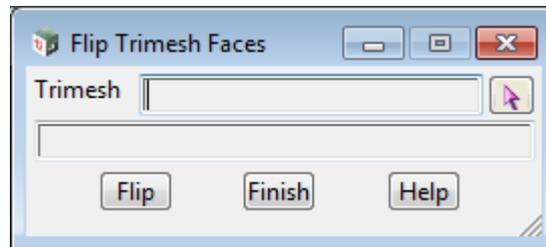
[17.6.6.1 Trimesh Flip Faces](#)

### 17.6.6.1 Trimesh Flip Faces

Position of option on menu: **Strings =>Trimesh =>Tools =>Flip Trimesh Faces**

This option reverses the order of the vertices in all the triangle in a trimesh.

Selecting **Flip faces** brings up the **Flip Trimesh Faces** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Trimesh</b> <i>trimesh to flip the faces.</i>	trimesh select		
<b>Flip</b> <i>reverse the order of the vertices in all the triangles in the selected trimesh.</i>	button		

Return to [17.6 Trimesh](#).

## 17.6.7 Generate Trimesh from 12d Objects

The option is currently at

**Strings =>Trimesh =>Generate trimesh from 12d Objects**

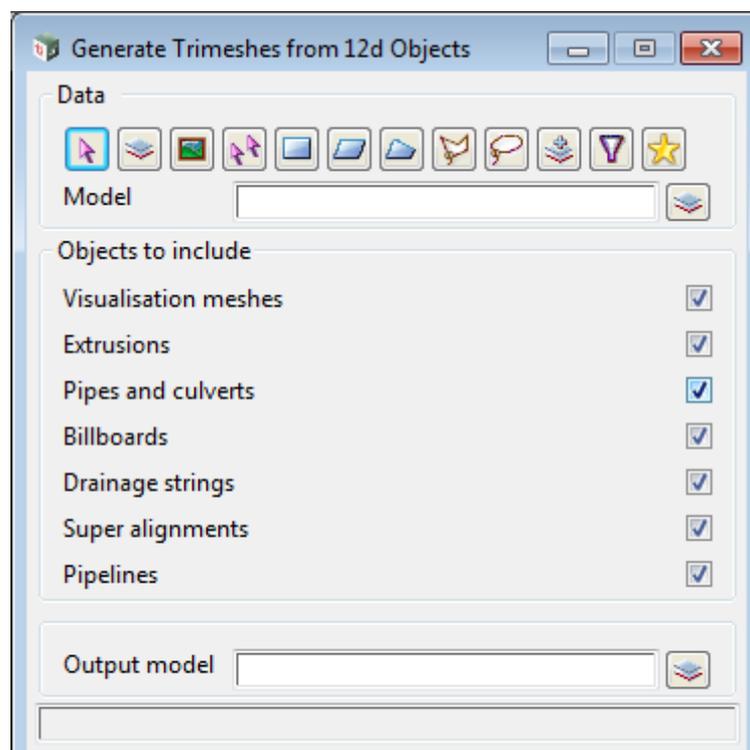
This option creates trimeshes for any mesh, extrude, super string pipe or culvert, drainage/sewer string or billboard, in the selected data.

The trimeshes can then be added to a section view, or cross section plots, to see that they are correctly sectioned through.

### IMPORTANT NOTE

All the triangles in a trimesh currently all have the one colour.

Selecting **Generate trimesh from 12d Objects** brings up the **Generate Trimeshes from 12d Objects** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b> <i>data selection type.</i>		Model	
<b>Data source</b> <i>source of data to process. For any meshes, a trimesh is created.</i>	input		
<b>Visualisation meshes</b> <i>if ticked, any meshes in the data source are converted to trimeshes. If not ticked, then meshes are not converted to trimeshes.</i>	tick box	tick	
<b>Extrusions</b> <i>if ticked, any extrudes in the data source are converted to trimeshes. If not ticked, then extrudes are not converted to trimeshes.</i>	tick box	tick	

- Pipes and culverts**            tick box            tick  
*if **ticked**, any super string pipe or culvert in the data source are converted to trimeshes.  
If **not ticked**, then super string pipe or culverts are not converted to trimeshes.*
- Billboard**                    tick box            tick  
*if **ticked**, any billboard in the data source are converted to trimeshes.  
If **not ticked**, then billboards are not converted to trimeshes.*
- Drainage strings**            tick box            tick  
*if **ticked**, any drainage/sewer strings in the data source are converted to trimeshes.  
If **not ticked**, then drainage/sewer strings are not converted to trimeshes.*
- Output model**                model box                            available models  
*model to add all the created trimeshes to.*
- Generate**                    button  
*create trimeshes for all the selected type that are in the data source.*

## 17.6.8 Generate Trimesh from Tin

**Position of option on menu:** Strings =>Trimesh =>Generate trimesh from tin

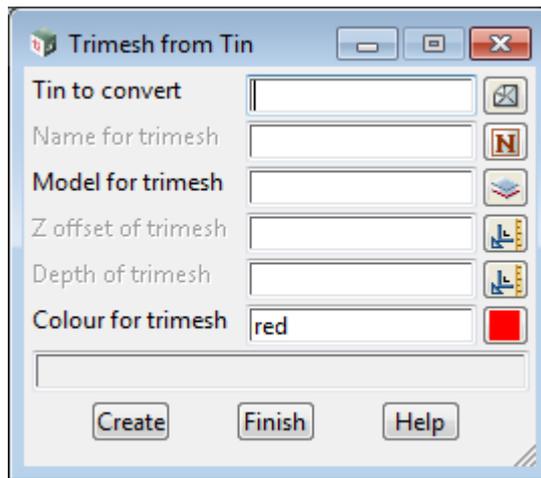
This option takes a tin and creates a trimesh of a user defined thickness from the tin.

The trimesh can also be offset in z from the tin by a user defined value.

### IMPORTANT NOTE

All the triangles in a trimesh currently have just the one colour.

Selecting **Generate trimesh from tin** brings up the **Generate Trimesh from tin** panel.



The fields and buttons used in this panel have the following functions.

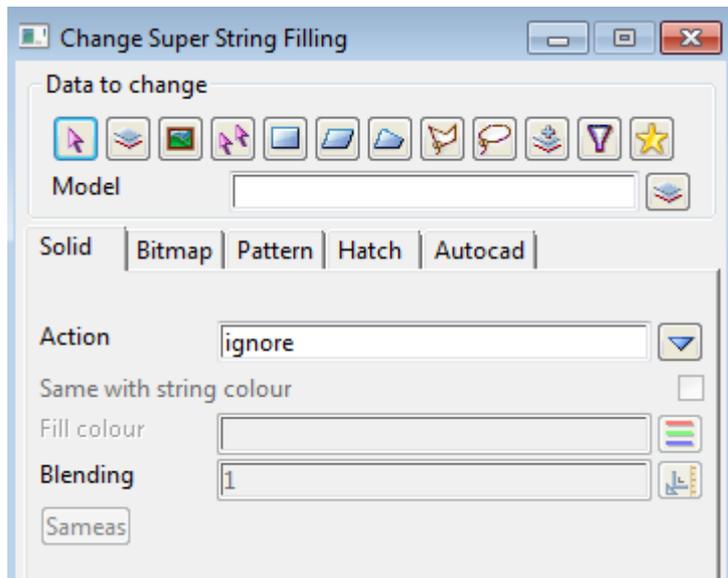
Field Description	Type	Defaults	Pop-Up
<b>Tin to convert</b> <i>tin to create a trimesh from.</i>	tin box		
<b>Name for trimesh</b> <i>name for the created trimesh.</i>	name box		
<b>Model for trimesh</b> <i>model to add the created trimesh to.</i>	model box		available models
<b>Z offset of trimesh</b> <i>the top of the trimesh is the tin translated up by this value. If the value is negative then the trimesh is lowered rather than raised.</i>	real box		
<b>Depth Z offset of trimesh</b> <i>the bottom of the trimesh is this value below the top of the trimesh. If the value is negative then the "bottom" of the trimesh is above the "top" of the trimesh.</i>	real box		
<b>Colour for trimesh</b> <i>colour for the created trimesh.</i>	colour box		available colours
<b>Create</b> <i>create trimesh from the tin.</i>	button		

## 17.7 Super String Fills

**Position of option on menu:** Strings =>Properties =>Super string dimensions =>Fills

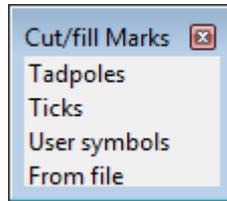
On the **Solid** tab of the **Change Super String Filling** panel, there is now a **Same as string colour** tick box.

If it is ticked **on** then for all the selected strings that are to have their **Solid fills** modified, the colour of the super string solid fill is set to be the same as the string colour.



# 17.8 Strings =>Label =>Cut/fill

Position of option on menu: Strings =>Label =>Cut/fill



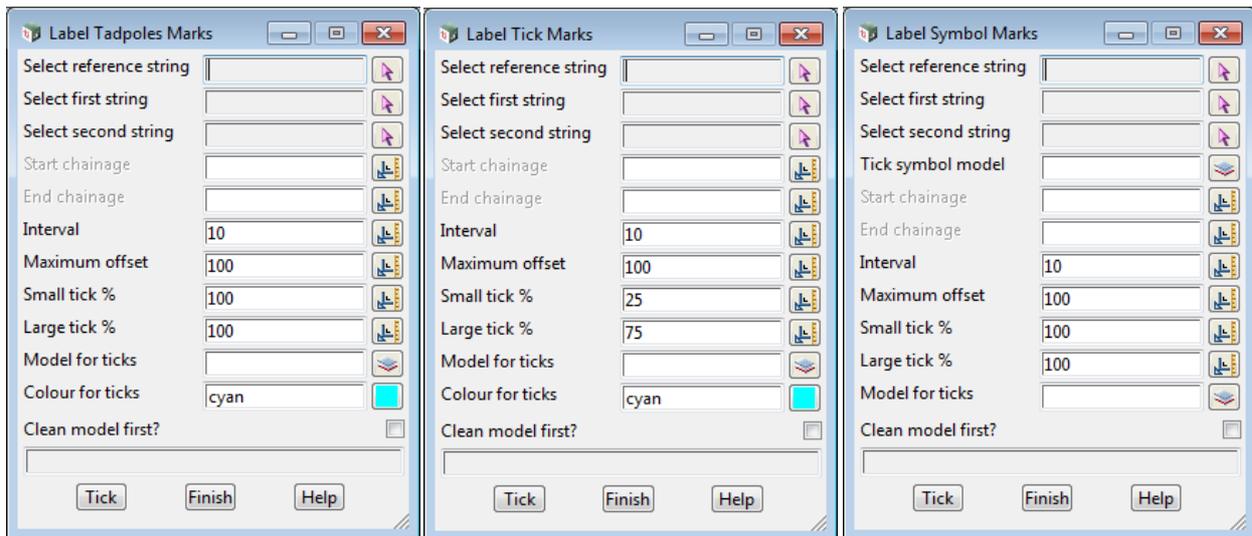
The three options **Tadpoles**, **Ticks** and **User Symbols** on this menu now have a tick box

**Clean model first ?**

and if it is ticked, the model **Model for ticks** is cleaned before the option runs.

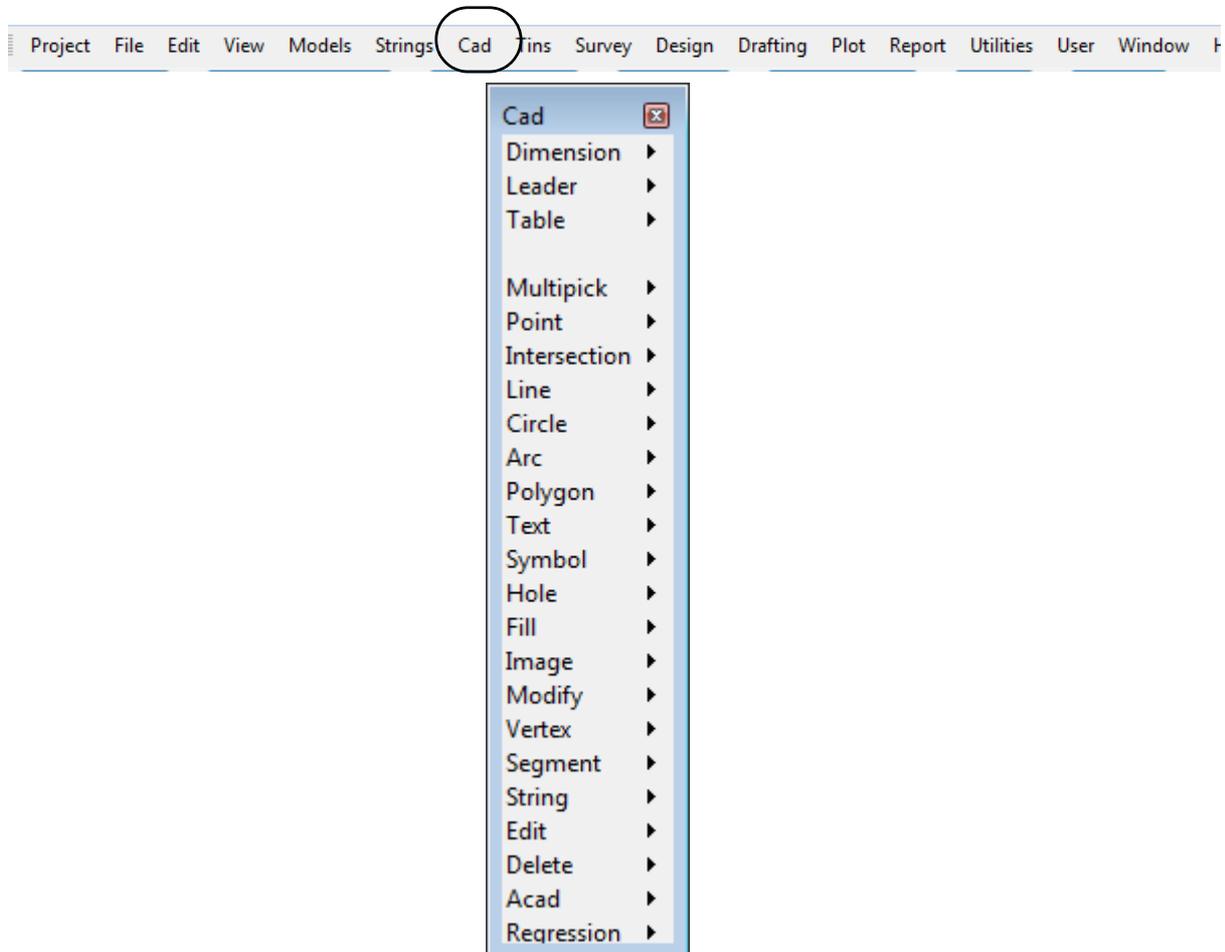
The panels **Label Tadpoles Marks** and **Label Symbol Marks** also have **String select** boxes for picking strings rather than the buttons at the bottom that were used in V10.

This means that all three panels can be recorded in chains.



# 18. Cad Menu

The CAD menu has been taken out of the Strings menu and is now a top menu called **Cad**.



See

[18.1 CAD Dimension](#)

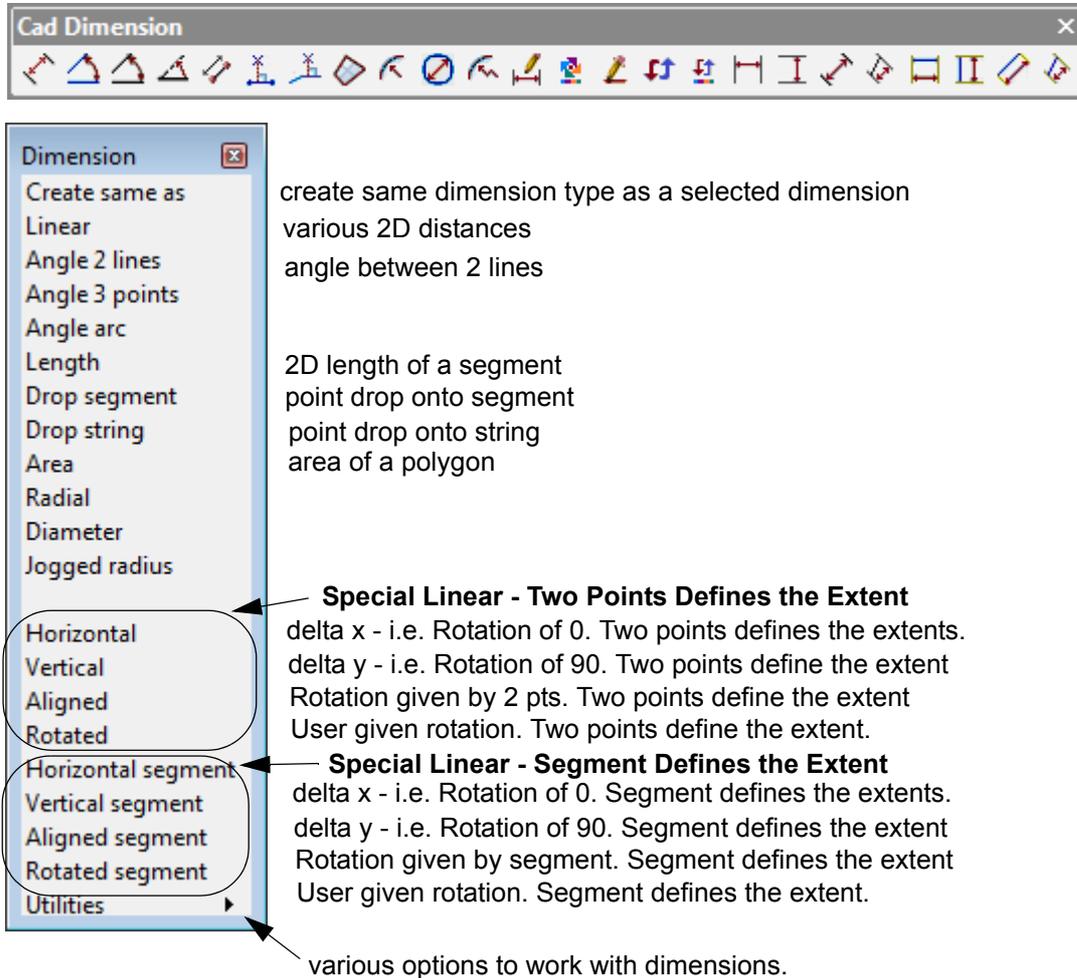
[18.2 CAD Leader](#)

[18.4 CAD Table](#)

# 18.1 CAD Dimension

**Position of option on menu: CAD =>Dimension**

Various types of **Dimensions** can be created with the **Dimension** options.



Each type of **Dimension** remembers the items it was created from (Association). In most cases when the items are modified, the **Dimensions** and the associated values on the Dimension dynamically change. See [18.1.1 Dimension Association](#).

For information about the meaning of the linear dimension commands Linear, Horizontal, Vertical, Aligned, Rotated, Horizontal segment, Vertical segment, Aligned segment and Rotated segment see

For information about the meaning of the linear dimension commands Linear, Horizontal, Vertical, Aligned, Rotated, Horizontal segment, Vertical segment, Aligned segment and Rotated segment see [18.1.6 Linear Dimension Commands](#).

For more information on each of the dimension commands, see

**Create same as** - create new dimension of same type as selected dimension [18.1.5 Dimension - Create Same As](#)

**Linear** - various 2D distances. See [18.1.6.1 Linear](#)

**Angle 2 lines** - angle between two selected lines. See [18.1.14 Angle 2 Lines](#)

**Angle 3 points** - angle between two lines defined by 3 points. See [18.1.15 Angle 3 points](#)

**Angle Arc** - angle subtended by existing arc or a segment with a radius. See [18.1.16 Angle Arc](#)

**Length** - the 2D length of selected segments See [18.1.7 Length](#)

**Drop segment** - point dropped onto segment. See [18.1.8 Drop Segment](#)

**Drop string** - point dropped onto a string. See [18.1.9 Drop String](#)

**Area** - area of selected strings. See [18.1.10 Area](#)

**Radius** - radius of a selected arc. See [18.1.13 Radius](#)

**Diameter** - diameter of an arc [18.1.12 Diameter](#) .

**Jogged radius** - radius of a selected arc with jog. See [18.1.11 Jogged radius](#)

**Arc Length - Arc by Centre, 2 Points** - length of an arc with arc defined by selecting a centre and two points on the arc. See [18.1.17 Arc Length by Centre, 2 Points](#)

**Horizontal** - the horizontal distance (delta x) between two selected points. See [18.1.6.3 Horizontal](#)

**Vertical** - the vertical distance (delta y) between two selected points. See [18.1.6.4 Vertical](#)

**Aligned** - 2D distance between two selected points. See [18.1.6.2 Aligned](#)

**Rotated** - the 2D between a selected point and another selected point dropped onto the line going through the first point with a user given angle. See [18.1.6.5 Rotated](#)

**Horizontal segment** - the horizontal distance (delta x) between the end vertices of selected segments. See [18.1.6.7 Horizontal Segment](#)

**Vertical segment** - the vertical distance (delta y) between the vertical points of the selected segment. See [18.1.6.8 Vertical Segment](#)

**Aligned segment** - the 2D distance between the end vertices of selected segments. See [18.1.6.6 Aligned Segment](#)

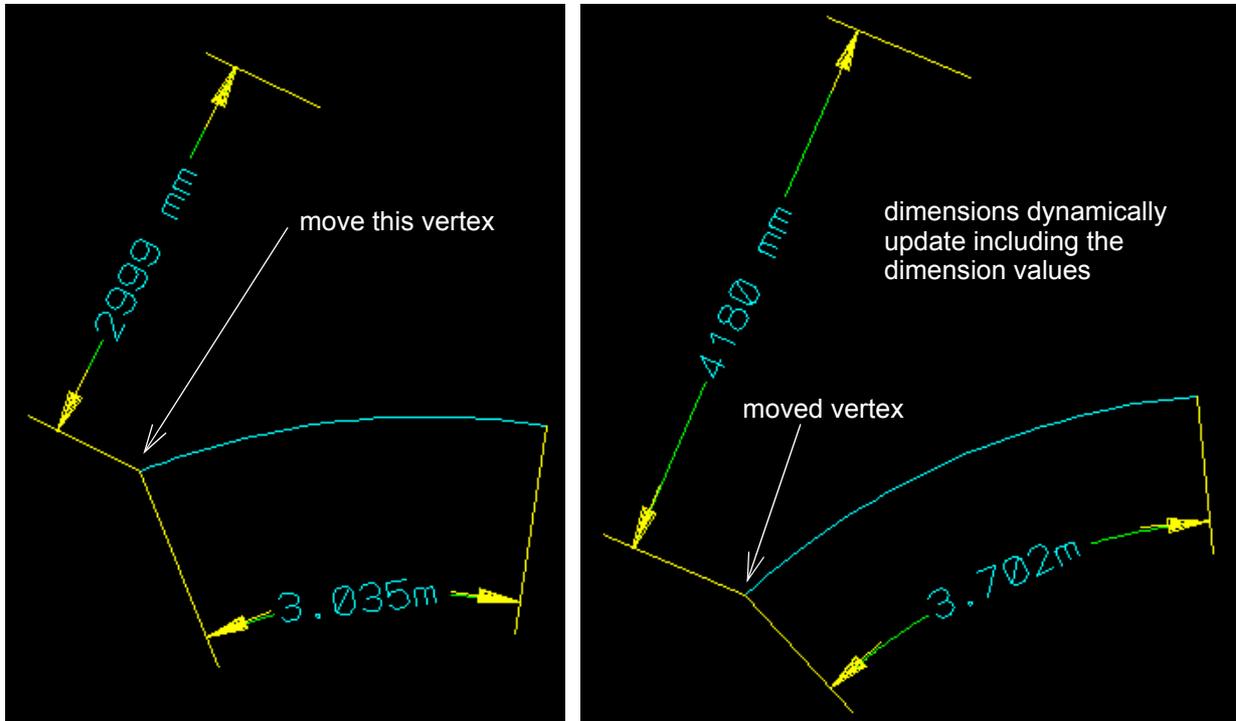
**Rotated segment** -the 2D between the first vertex of a selected segment and the end vertex of the segment dropped onto the line going through the first vertex with a user given angle. See [18.1.6.9 Rotated Segment](#)

**Utilities** - various dimension utilities. See [18.1.18 Dimension Utilities](#)

## 18.1.1 Dimension Association

When *dimensions* are created, the dimension can remember the items that were used to create them (**Association**).

In most cases when items are moved, the dimensions associated with the items dynamically change.

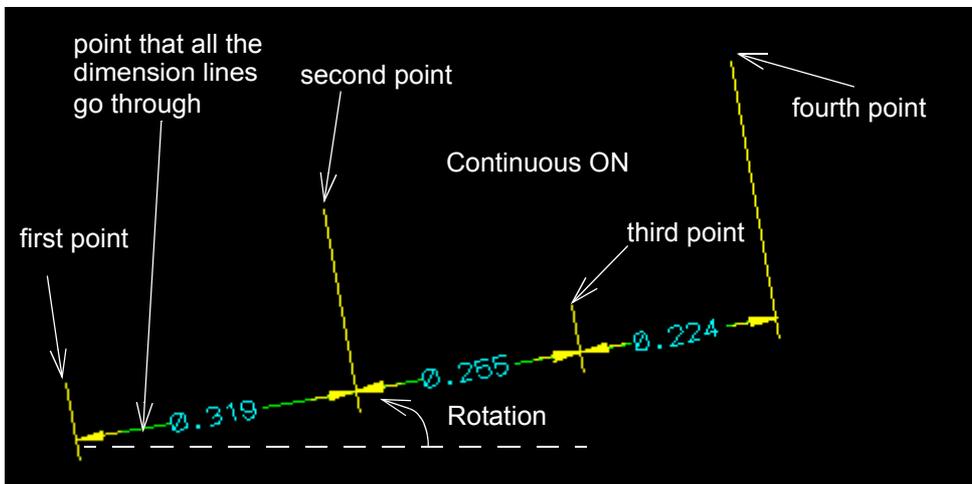
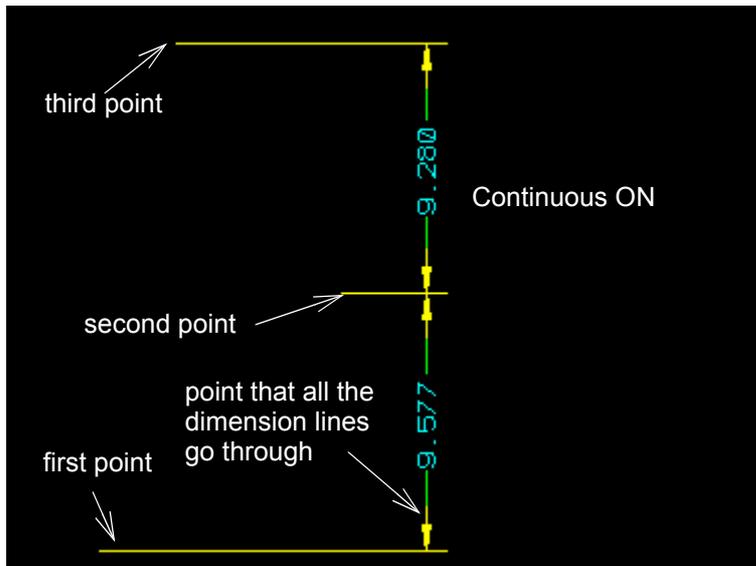
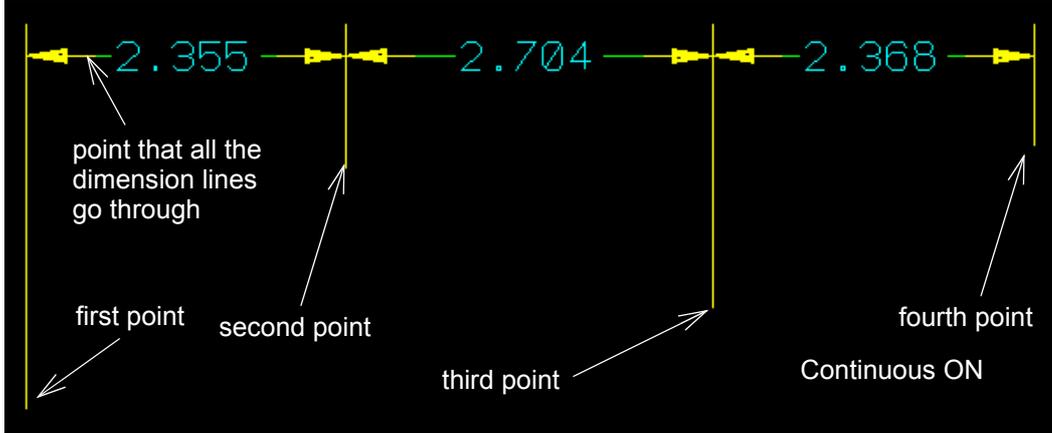


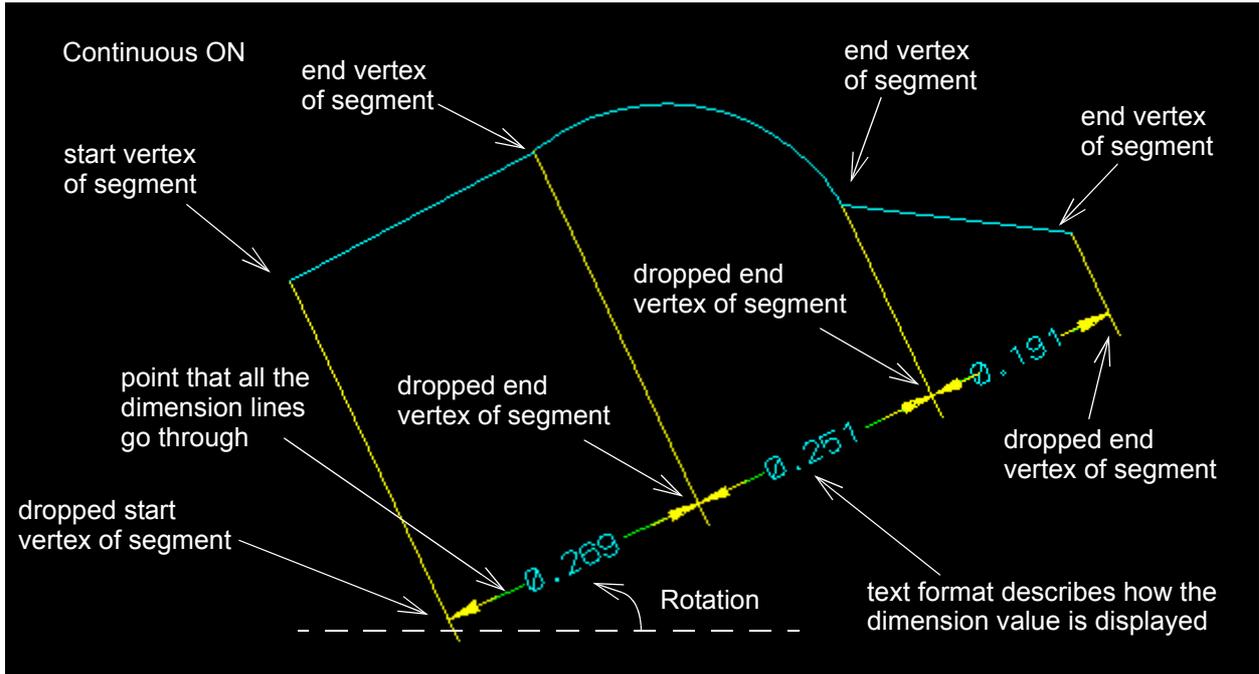
In cases where it is not automatic, there is a **Recalc** and **Recalc all** for dimensions and leaders. In the dimension options, typing **a** or **A** toggles Association **ON** and **OFF**.

## 18.1.2 Continuous Mode

For all the **Linear** dimension options except **Aligned**, there is a **Continuous** mode.

When Continuous mode is **on**, all the dimensions lines are on the same line as the dimension line of the first dimension done in the sequence.





In the dimension options which support Continuous, typing **c** or **C** toggles Continuous **ON** and **OFF**.

## 18.1.3 Baseline Mode

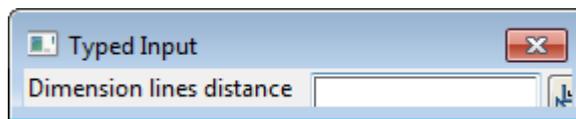
For all the dimension options *Horizontal*, *Vertical* and *Rotated*, there is a **Baseline** mode.

When **Baseline** mode is **ON**, the distances are all measured from the first picked point, the first dimension line goes through the picked dimension point, and for each subsequent dimension line, the distance is measured from the first point, and the dimension line is moved a give distance offset to the previous dimension line so that the dimension lines for not overlap.

In the dimension options which support *Baseline*, **(b)aseline** appears in the message in the screen message area:

```
<[(a)ssociation ON] [(c)ontinuous OFF] [(b)aseline OFF] Pick start point or choose (s)tyle[d1] > [picks][accepts][Menu]
```

Typing **b** or **B** toggles **Baseline ON** and **OFF** and when **Baseline** is toggle **ON** for the first time, the *Dimension lines distance* **Typed Input** box is place on the screen.



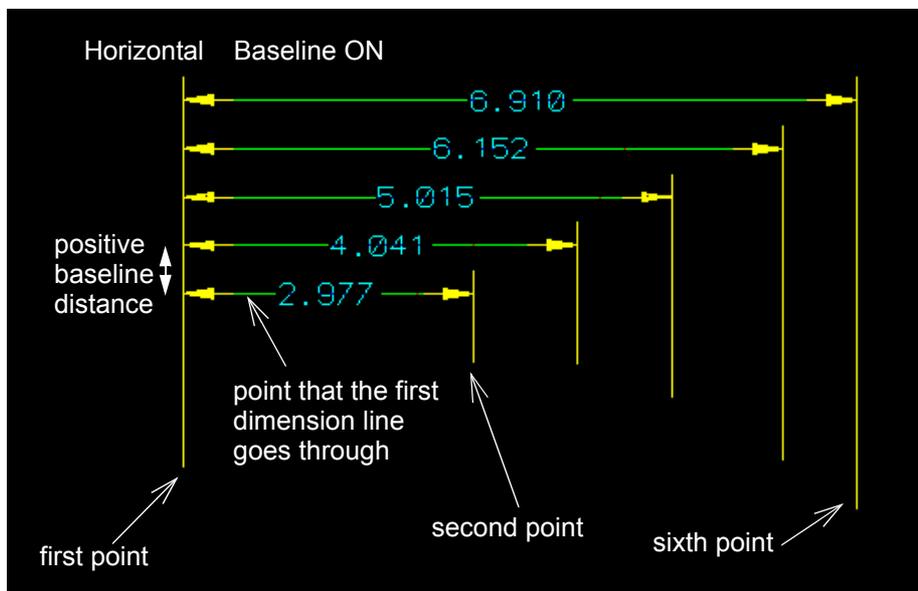
The new distance is typed into the box and <Enter> pressed.

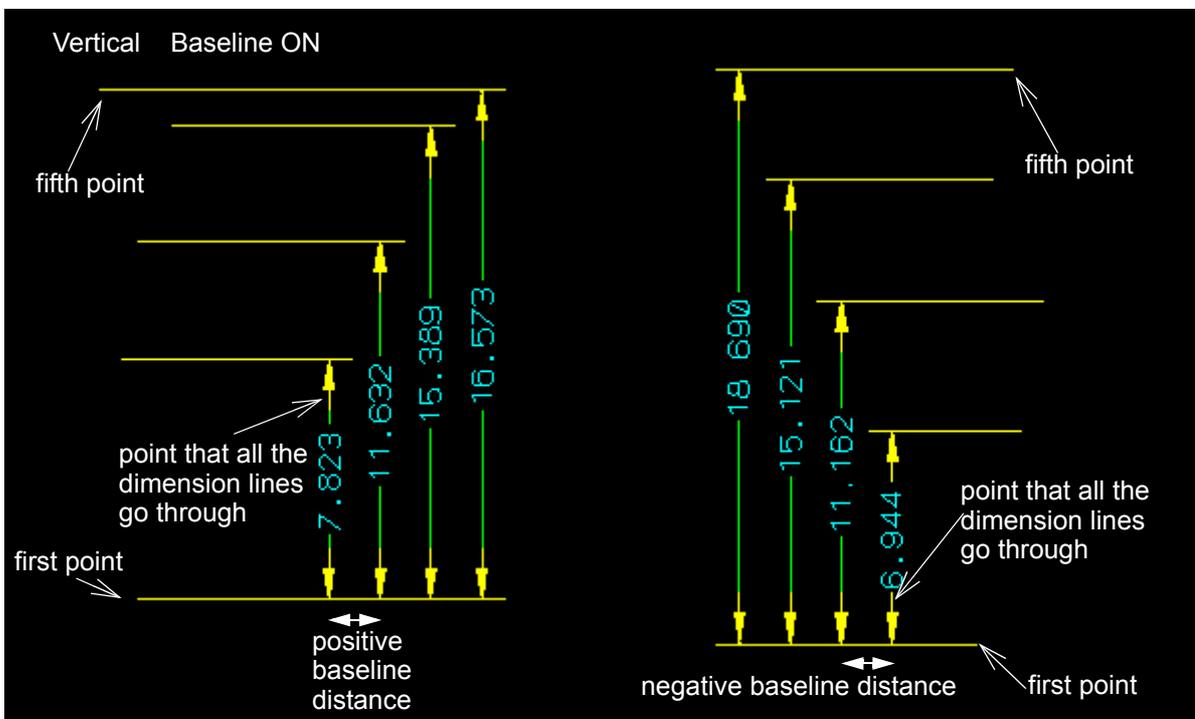
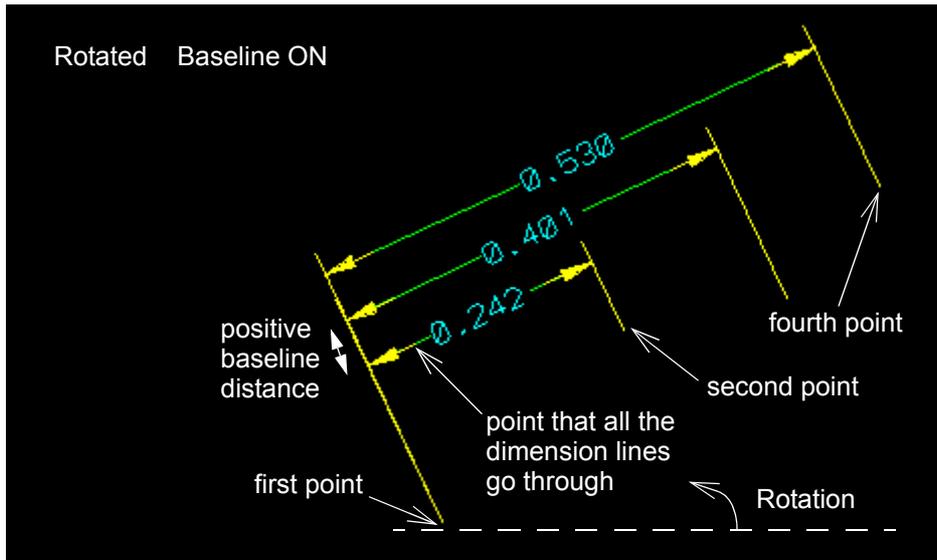
Whenever *Baseline* is toggled **ON**, **(d)istance** appears in the message in the screen message area:

```
<[(a)ssociation ON] [(c)ontinuous OFF] [(b)aseline ON (d)istance[0.500]]Pick start point or choose (s)tyle[d1] > [picks]
```

Typing **d** or **D** toggles brings up the *Dimension lines distance* **Typed Input** box and a new distance can be entered.

Whenever **Baseline** is **ON**, **Continuous** is turned **OFF**.





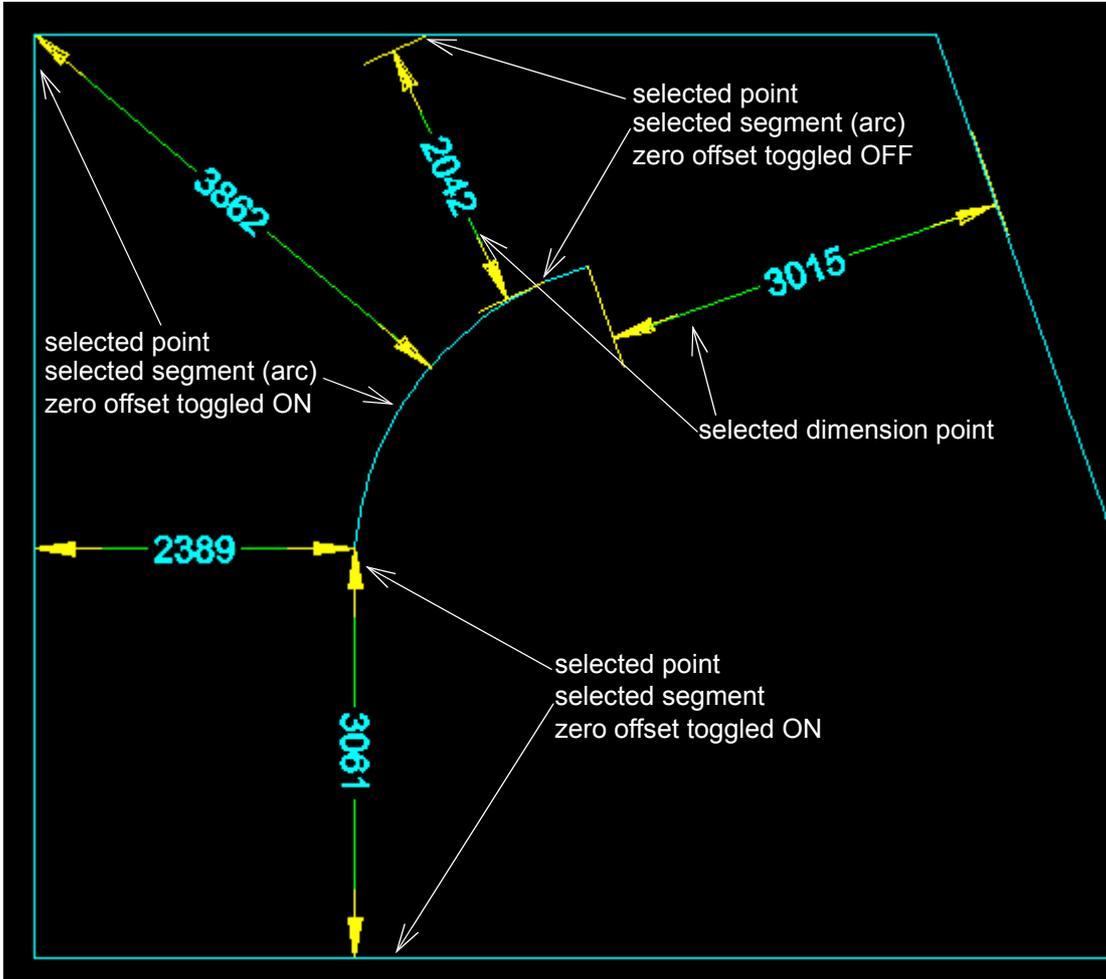
## 18.1.4 Zero Offset Mode

For some dimension options there is a **Zero Offset** mode.

When **Zero Offset** mode is **ON**, then the user is not asked to select a dimension point.

The dimension line is automatically placed on the line between the two points

When **Zero Offset** is available, typing **z** or **Z** toggles **Zero offset ON** and **OFF**.



## 18.1.5 Dimension - Create Same As

**Create same as** creates a new Dimension of the same type as a selected Dimension.

### Step 1.

Select **Create same as** and the following message is written to the screen message area

```
<[Pick dimension]> [picks][fast][Menu]
```

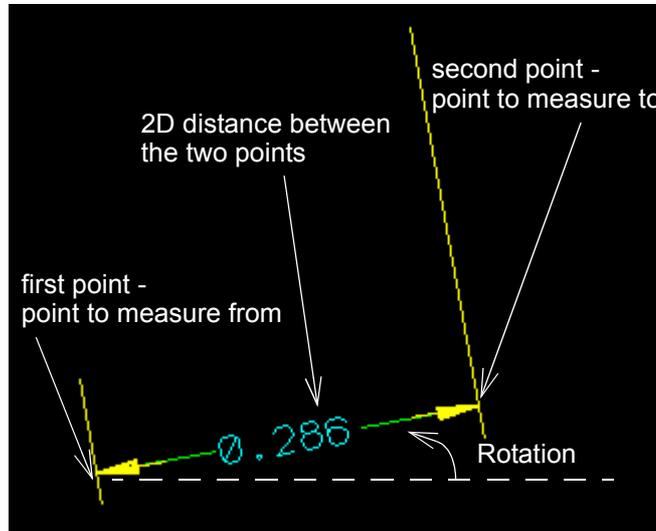
### Step 2. Select an existing Dimension

When an existing **Dimension** is selected, a new **create Dimension** of the same Dimension type as the selected Dimension, is started.

## 18.1.6 Linear Dimension Commands

The **Linear Dimension** commands measure the **2D (plan) distance between two points**.

The two points to measure between define a line and that line has a **Rotation** angle with respect to the x-axis (Rotation is measured in an anticlockwise direction from the positive x-axis).



Note that the two points may be selected individually or can be the end points of a selected segment (if the segment is an arc or a transition, only the end points of the segment are used).

Because most drawings to be dimensioned are very regular and often drawn so that the major line work is parallel to the x and y axes, it is an advantage to have special forms of the **Linear Dimension** so that the intent is clearer and less information has to be provided by the user.

The **special Linear Dimensions** are best classified by their **Rotation** angle.

(a) **Horizontal - rotation angle of 0**

This is the difference between the two x coordinates of the points or segment (delta x).

The Rotation is known (0) so once the first point is selected, the software automatically drops the second point onto the line with Rotation 0 that goes through the first point and uses the x value from the dropped point.

**Horizontal** and selecting **two points**.

See [18.1.6.3 Horizontal](#)

**Horizontal** and selecting a **segment**.

See [18.1.6.7 Horizontal Segment](#)

(b) **Vertical - rotation angle of 90**

This is the difference between the two y coordinates (delta y).

The Rotation is known (90) so once the first point is selected, the software automatically drops the second point onto the line with Rotation 90 that goes through the first point and uses the y value from the dropped point.

**Vertical** and selecting **two points**.

See [18.1.6.4 Vertical](#)

**Vertical** and selecting a **segment**

See [18.1.6.8 Vertical Segment](#)

(c) **Rotated - a user given rotation value.**

The Rotation is known so once the first point is selected, the software automatically drops the second point onto the line with the given Rotation that goes through the first point and uses the dropped point to measure the 2D distance to.

**Rotated** and selecting **two points**.

See [18.1.6.5 Rotated](#)

**Rotated** and selecting a **segment**

See [18.1.6.9 Rotated Segment](#)

(d) **Aligned - rotation defined by selecting two points or a segment**

This is a more general case where the two points have to accurately selected and are used to defined the Rotation. This is the only case when the Rotation is not known before selecting anything.

**Aligned** and selecting two points.

See [18.1.6.2 Aligned](#)

**Aligned** and selecting a **segment**

See [18.1.6.6 Aligned Segment](#)

Hence in **12d Model** there are **eight separate commands** for defining a Linear dimension with a special rotation, and **one** general **Linear** command that covers all **eight Linear command**.

See

**Linear** - all eight cases.

See [18.1.6.1 Linear](#)

### 18.1.6.1 Linear

The **Linear** option allows the use any of the eight modes of defining a Linear dimension.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

#### Step 1.

Select **Linear** and the following message is written to the screen message area

```
<[Pick start poi(n)t] [(h)orizontal (v)l(r)] [(c)ontinuous OFF] [(b)aseline OFF] (a)ssociative (s)tyl[e Arial Aligned] forma(t) > [picks][fast][Menu]
```

#### Step 2.Settings

Typed one characters commands are now required and the first ones to set are those that don't actually start a dimension but are set ups - **n**, **a**, **c**, **s**.

Some of the commands, **n**, **a**, **c**, **s**

**n** or **N** toggles between picking two points or picking a segment.

```
<[Pick start poi(n)t] [(h)orizontal (v)l(r)] [(c)ontinuous OFF] (a)ssociative (s)tyl[e Arial Aligned] forma(t) > [picks][fast][Menu]
<[Pick segme(n)t] [(h)orizontal (v)l(r)] [(c)ontinuous OFF] (a)ssociative (s)tyl[e Arial Aligned] forma(t) > [picks][fast][Menu]
```

clicking **n** toggles between point and segment

**a** or **A** toggles association **ON** and **OFF**.

**c** or **C** toggles continuous **ON** and **OFF**.

**s** or **S** brings up the dimension style list.

#### Step 3.Dimension Commands

The dimension commands are **horizontal**, **vertical**, **aligned** or **rotated** and the current **active** dimension command has square brackets around it.

**h** makes **horizontal** the active dimension command.

**v** makes **vertical** the active dimension command.

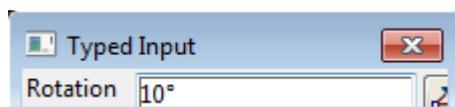
**l** makes **aligned** the active dimension command.

**r** makes **rotated** the active dimension command.

Note - When **rotated** is the active dimension command, the current value for the rotation angle is displayed in the message area.

```
[(h)(v)(l) (r)otated] (a)n(g)le[0°] rotation angle
```

Whenever the angle is displayed, typing **g** or **G** will bring up a **Typed Input** box to change the rotation angle.



#### Step 4.Creating a Dimension - picking first point or a segment.

What type of linear dimension is to be created has been set (h, v, l, r).

Whether it requires two points or a segment **has** been set.

Whether it will be **associated** or not, or **continuous** or not **has** been set.

Depending on the point/segment mode, the start of the message indicates whether picking a point or picking a segment is required. So now pick the first point or a segment.

How the **Linear** option operates is now identical to one of eight specialised linear dimension options so it will not be documented now but done in each of the specialised cases.

If **Horizontal** and **point**, see [18.1.6.3 Horizontal](#)

If **Horizontal** and **segment**, see [18.1.6.7 Horizontal Segment](#)

If **Vertical** and **point**, see [18.1.6.4 Vertical](#)

If **Vertical** and **segment**, see [18.1.6.8 Vertical Segment](#)

If **Aligned** and **point**, see [18.1.6.2 Aligned](#)

If **Aligned** and **segment**, see [18.1.6.6 Aligned Segment](#)

With **Rotated**, the angle is already set so you won't be asked for that again.

If **Rotated** and **point**, see [18.1.6.5 Rotated](#)

If **Rotated** and **segment**, see [18.1.6.9 Rotated Segment](#)

#### **Step 5.**

After the dimension has been created, the **Linear** option then starts again. That is, goes back to **Step 1**. You can continue with the current dimension type or switch to another dimension type by clicking on **h**, **v**, **l** or **r** (or **H**, **V**, **L** or **R**).

For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

### 18.1.6.2 Aligned

**Aligned** labels the 2D distance between two selected points.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

**z** or **Z** toggles *Zero Offset* mode **ON** and **OFF**. See [18.1.4 Zero Offset Mode](#)

#### Step 1. Pick the start point

Select **Aligned** and the following message is written to the screen message area

```
<[Pick start point] (a)ssociative (s)tyl[e][default] forma(t) [(z)ero offset OFF] > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected start point.

#### Step 2. Pick the end point

The following message is written to the screen message area.

```
<[Pick end point] (a)ssociative > [picks][fast][Menu]
```

If Association is **ON** then the end of dimension is associated with the selected end point.

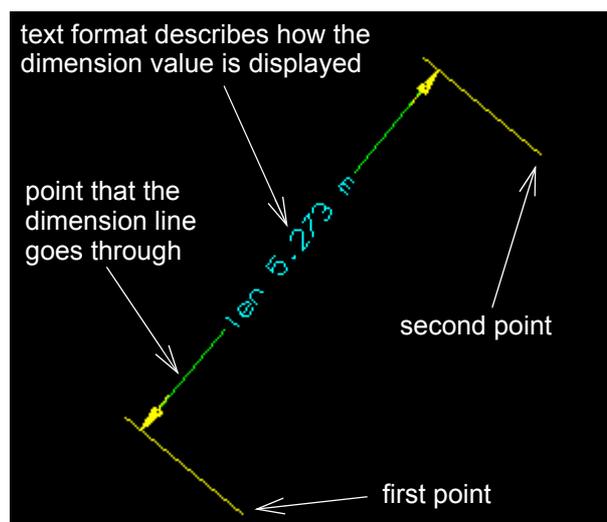
The **2D distance** is calculated between these two points.

#### Step 3. Pick the point that the dimension line will go through (dimension point)

The following message is written to the screen message area.

```
<Pick dimension point> [picks][fast][Menu]
```

The dimension line is then drawn parallel to the line between the two selected points and going through the dimension point.



The **Aligned** option then starts again. That is, goes back to **Step 1**.

For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

### 18.1.6.3 Horizontal

**Horizontal** labels the horizontal distance (delta x) between two selected points.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

#### Step 1. Pick the start point

Select **Horizontal** and the following message is written to the screen message area

```
<[Pick start point] [(c)ontinuous OFF] [(b)aseline OFF] (a)ssociative (s)tyle[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected first point.

#### Step 2. Pick the end point

The following message is written to the screen message area.

```
<[Pick end point] (a)ssociative > [picks][fast][Menu]
```

If Association is **ON** then the end of dimension is associated with the selected end point.

The horizontal distance (delta x) is calculated between these two points.

#### Step 3. Pick the point that the dimension line will go through (dimension point)

The following message is then written to the left hand side of the screen message area.

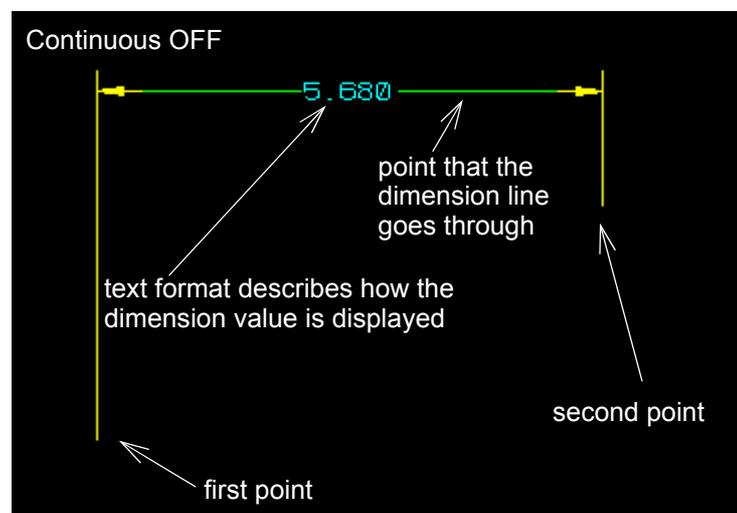
```
<[Pick dimension point] > [picks][fast][Menu]
```

The horizontal dimension line is then drawn parallel to the x-axis going through the dimension point.

#### Step 4.1

If Continuous is **OFF**

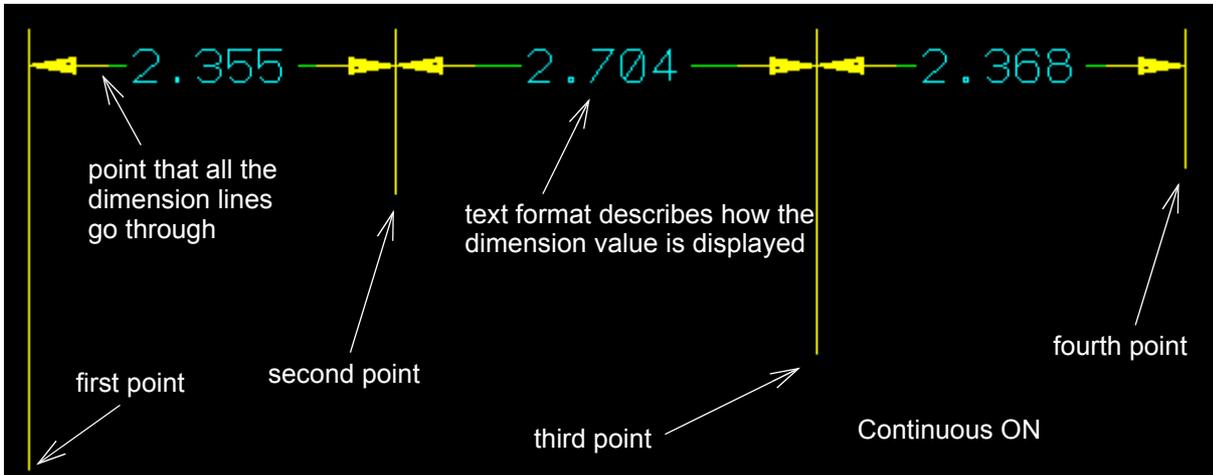
The **Horizontal** option then starts again. That is, goes back to **Step 1**.



If Continuous is **ON**

If Continuous is **ON** then you are asked to pick another end point and another Horizontal dimension is created with the start of the horizontal dimension being the previous dimension end, and the dimension point is the same as the previous dimension point.

This repeats until Continuous is toggle **OFF** and then the next selected end point is the last horizontal dimension in the sequence.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

### 18.1.6.4 Vertical

**Vertical** labels the vertical distance (delta y) between two selected points.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

#### Step 1. Pick the start point

Select **Vertical** and the following message is written to the screen message area

```
<[Pick start point] [(c)ontinuous OFF] [(b)aseline OFF] (a)ssociative (s)tyl[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected first point.

#### Step 2. Pick the end point

The following message is written to the screen message area.

```
<[Pick end point] (a)ssociative > [picks][fast][Menu]
```

If Association is **ON** then the end of dimension is associated with the selected end point.

The vertical length (delta y) is calculated between these two points.

#### Step 3. Pick the point that the dimension line will go through (dimension point)

The following message is then written to the left hand side of the screen message area.

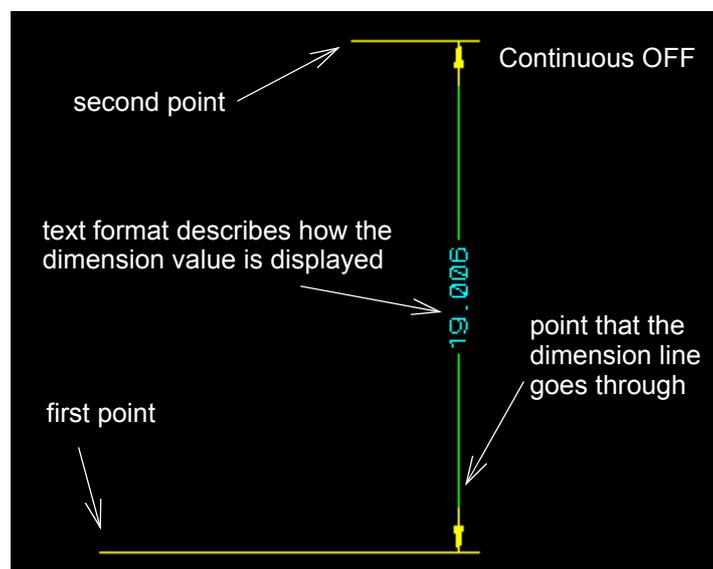
```
<[Pick dimension point] > [picks][fast][Menu]
```

The vertical dimension line is then drawn parallel to the y-axis going through the dimension point.

#### Step 4.1

If Continuous is **OFF**

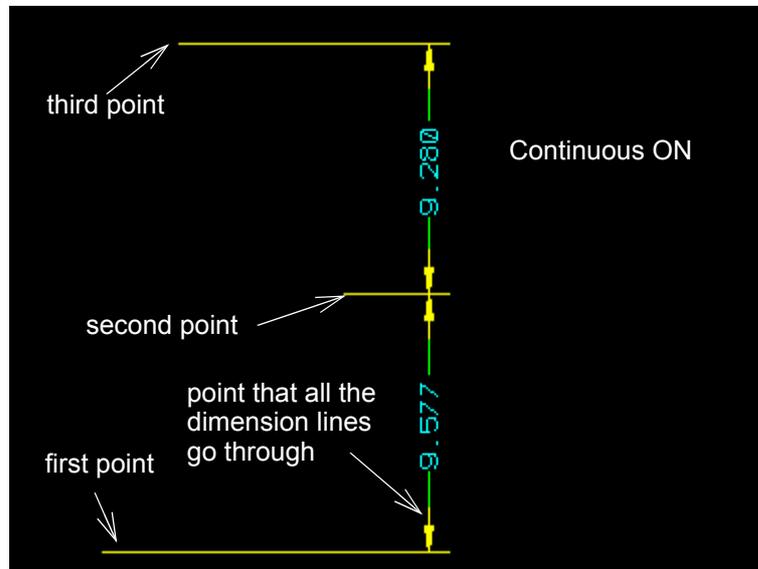
The **Vertical** option then starts again. That is, goes back to **Step 1**.



If Continuous is **ON**

If Continuous is ON then you are asked to pick another end point and another Vertical dimension is created with the start of the vertical dimension being the previous dimension end, and the dimension point is the same as the previous dimension point.

This repeats until Continuous is toggle **OFF** and then the next selected end point is the last vertical dimension in the sequence.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

### 18.1.6.5 Rotated

**Rotated** labels the **2D distance** between the first selected point and the second selected point **dropped** onto the line with a user given angle (Rotation), and going through the first selected point.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

#### Step 1. Pick the start point

Select **Rotated** and the following message is written to the screen message area

```
<[Pick start point] [(c)ontinuous OFF] [(b)aseline OFF] (a)ssociative (s)tyl[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected first point.

#### Step 2. Pick the end point

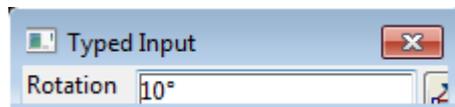
The following message is written to the screen message area.

```
<[Pick end point] (a)ssociative > [picks][fast][Menu]
```

If Association is **ON** then the end of dimension is associated with the selected end point.

#### Step 3. Type in the Rotation value

The following message is written to the screen message area.



The Rotation is in degrees in hp notation and is measured in an anticlockwise direction from the positive x axis.

#### Type in the Rotation value and <Enter>

When in **Continuous** mode the Rotation is only asked for when placing the first **Rotated** dimensions otherwise it is asked for each time.

**Note:** for the **Linear** command, the **Rotation** value is not asked for because it will have already been set.

#### Step 4. Pick the point that the dimension line will go through (dimension point)

The following message is then written to the left hand side of the screen message area.

```
<Pick dimension point> [picks][fast][Menu]
```

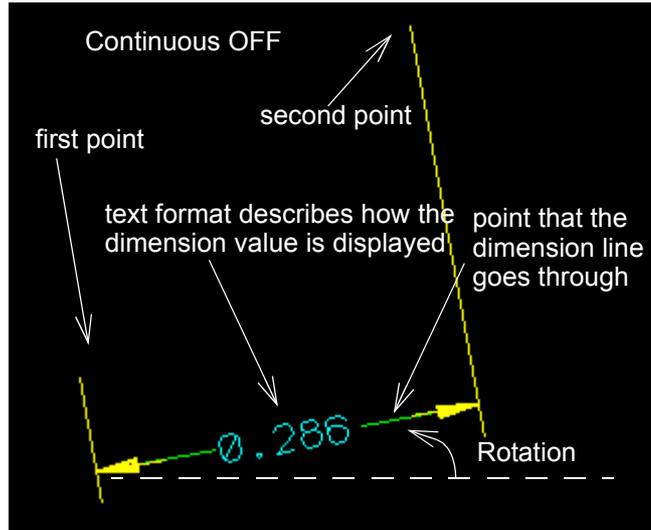
The **2D (plan) distance** is calculated between the first selected point and the second selected point dropped onto the line going through the first selected and with the user given angle (Rotation).

The dimension line is then drawn with the angle of the dimension line equal to the entered Rotation.

#### Step 5.1

If Continuous is **OFF**

The **Rotated** option then starts again. That is, goes back to **Step 1**.

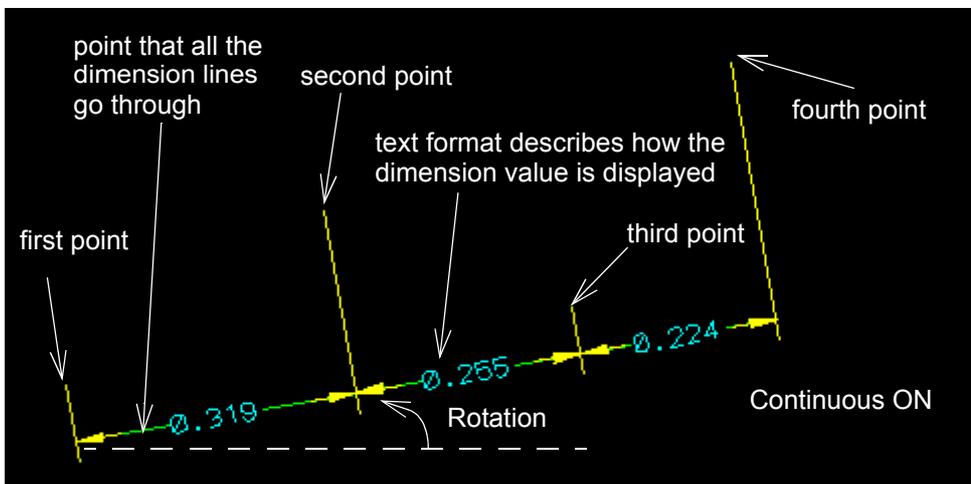


If Continuous is **ON**

If Continuous is ON then you are asked to pick another end point and another Rotated dimension is created with the start of the Rotated dimension being the previous dimension end, and the dimension point is the same as the previous dimension point.

The **2D (plan) distance** is calculated between the previous dropped selected point and the new selected point dropped onto the line with the user given angle (Rotation) going through the first selected point.

This repeats until Continuous is toggle OFF and then the next selected end point is the last Rotated dimension in the sequence.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles.](#)

### 18.1.6.6 Aligned Segment

**Aligned segment** labels the **2D distance** between the start and end vertices of a selected segment. The segment can be straight, arc or a transition.

The dimension is added to the model given in the CAD toolbar.

Note that this is **not** the same as the length of a segment for an arc, a transition or offset transition.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#).

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#).

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#).

#### Step 1. Pick the segment

Select **Aligned segment** and the following message is written to the screen message area

```
<[Pick segment] (a)ssociative (s)tyle[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the selected segment.

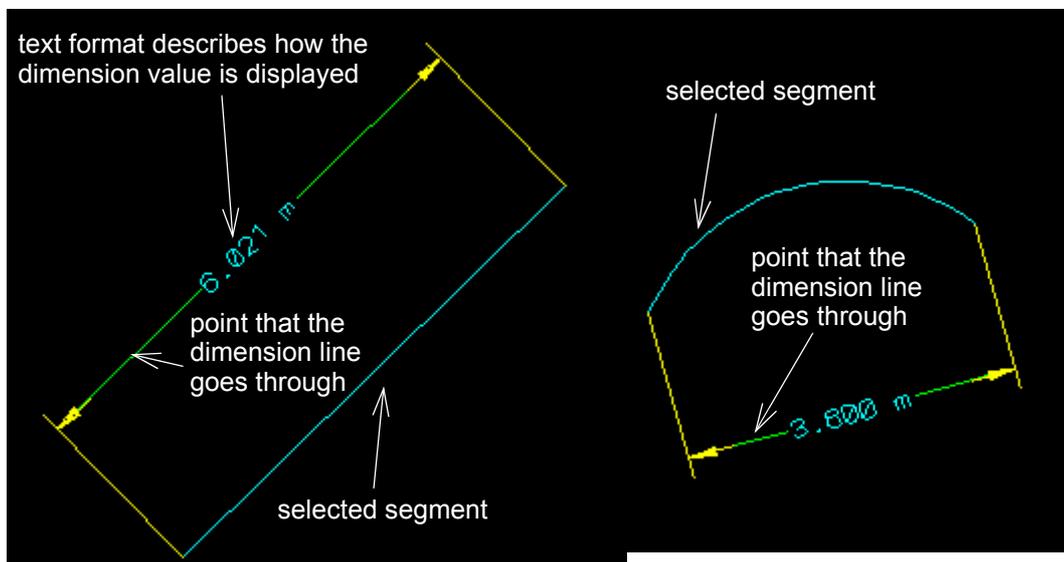
The 2D distance is calculated between the start and end vertices of the selected segment.

#### Step 2. Pick the point that the dimension line will go through (dimension point)

The following message is then written to the left hand side of the screen message area.

```
<Pick dimension point> [picks][fast][Menu]
```

The dimension line is then drawn parallel to the line from the start to the end vertices of the segment and going through the dimension point.



The **Aligned segment** option then starts again. That is, goes back to **Step 1**.

For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

### 18.1.6.7 Horizontal Segment

**Horizontal segment** labels the horizontal distance (delta x) between the start and end vertices of a selected segment. The segment can be straight, arc or a transition.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#).

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#).

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#).

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#).

#### Step 1. Pick the segment

Select Horizontal segment and the following message is written to the screen message area

```
<[Pick segment] (a)ssociative (s)tyl[e[default] forma(t) [(c)ontinuous OFF] > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the selected segment.

The horizontal distance (delta x) is calculated between the start and end vertices of the selected segment.

#### Step 2. Pick the point that the dimension line will go through (dimension point)

The following message is then written to the screen message area.

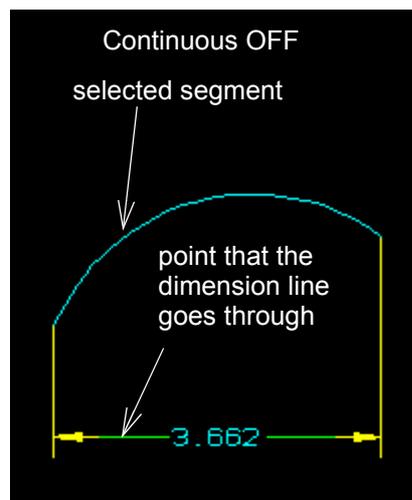
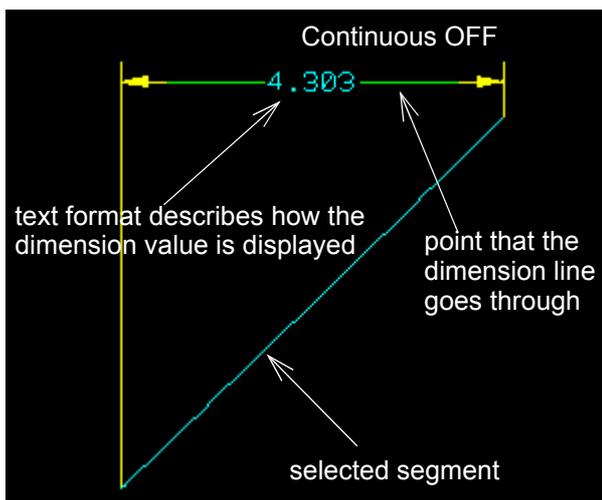
```
<Pick dimension point> [picks][fast][Menu]
```

The horizontal dimension line is then drawn parallel to the x-axis going through the dimension point.

#### Step 3. I

If Continuous is **OFF**

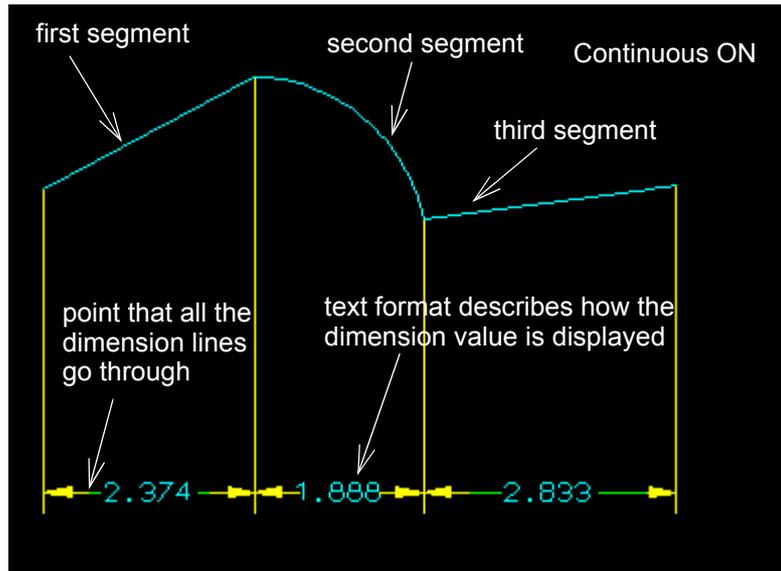
The **Horizontal segment** option then starts again. That is, goes back to **Step 1**.



If Continuous is **ON**

If Continuous is **ON** then you are asked to pick another segment and another Horizontal segment dimension is created for the next segment and the dimension point is the same as the previous dimension point.

This repeats until Continuous is toggle **OFF** and then the next selected segment is the last horizontal dimension in the sequence.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

### 18.1.6.8 Vertical Segment

**Vertical segment** labels the vertical distance (delta y) between end vertices of a selected segment. The segment can be straight, arc or a transition.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#).

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#).

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#).

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#).

#### Step 1. Pick the segment

Select Vertical segment and the following message is written to the screen message area

```
<[Pick segment] (a)ssociative (s)tyle[default] forma(t) [(c)ontinuous OFF] > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the selected segment.

The vertical distance (delta y) is calculated between the start and end vertices of the selected segment.

#### Step 2. Pick the point that the dimension line will go through (dimension point)

The following message is then written to the screen message area.

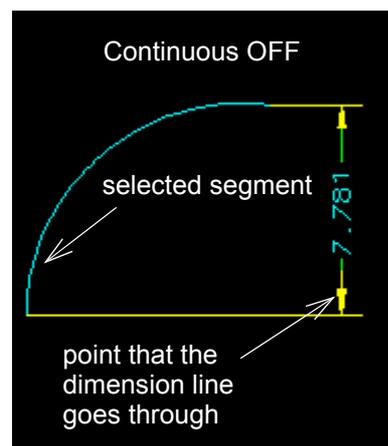
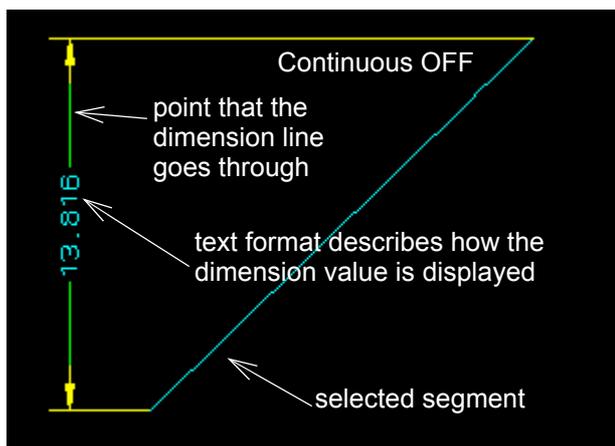
```
<Pick dimension point> [picks][fast][Menu]
```

The vertical dimension line is then drawn parallel to the y-axis going through the dimension point.

#### Step 3. I

If Continuous is **OFF**

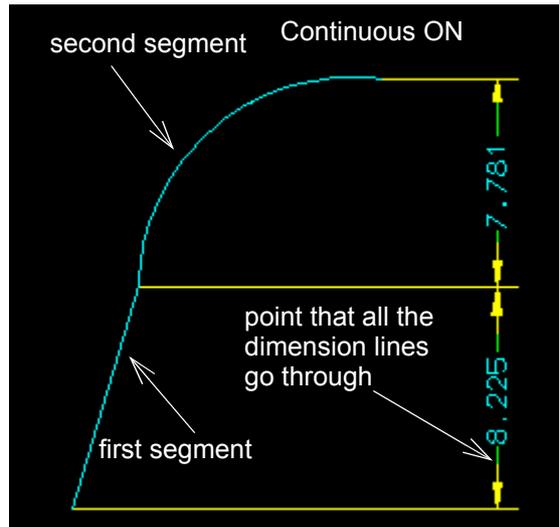
The **Vertical segment** option then starts again. That is, goes back to **Step 1**.



If Continuous is **ON**

If Continuous is **ON** then you are asked to pick another segment and another Vertical segment dimension is created for the next segment and the dimension point is the same as the previous dimension point.

This repeats until Continuous is toggle **OFF** and then the next selected segment is the last vertical dimension in the sequence.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#) .

### 18.1.6.9 Rotated Segment

**Rotated segment** labels the 2D distance between the start vertex of a segment, and the end vertex of the segment dropped onto the line with a user given angle (Rotation) and going through the first vertex.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

#### Step 1.

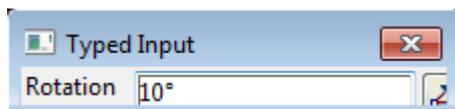
Select **Rotated segment** and the following message is written to the screen message area

```
<[Pick segment] (a)ssociative (s)tyle[default] forma(t) [(c)ontinuous OFF] an(g)le[10°] > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the selected segment.

#### Step 2.Type g to Change the Rotation Value

If the Rotation is to be changed, type **g** or **G** and the **Rotation Typed Input** box is then displayed.



**Type in the Rotation value and press <Enter>.**

The Rotation is in degrees in hp notation and is measured in an anticlockwise direction from the positive x axis.

#### Step 3.Pick the segment

The following message is written to the screen message area.

```
<[Pick segment] (a)ssociative (s)tyle[default] forma(t) [(c)ontinuous OFF] an(g)le[10°] > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the selected segment.

#### Step 4.Pick the point that the dimension line will go through (dimension point)

The following message is then written to the screen message area.

```
<Pick dimension point> [picks][fast][Menu]
```

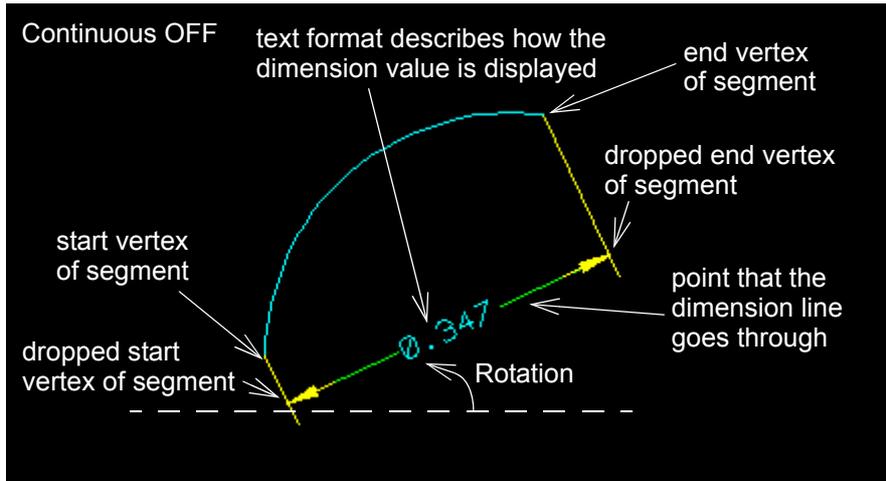
The **2D (plan) distance** is calculated between the start of the segment and the end of the segment dropped onto the line with the user given angle (Rotation) going through the start of the selected segment.

The dimension line is then drawn with the angle of the dimension line equal to the entered **Rotation**.

#### Step 5.

If Continuous is **OFF**

The **Rotated** option then starts again. That is, goes back to **Step 1**.



If Continuous is **ON**

If Continuous is ON then you are asked to pick another segment. The Rotation angle can also be changed.

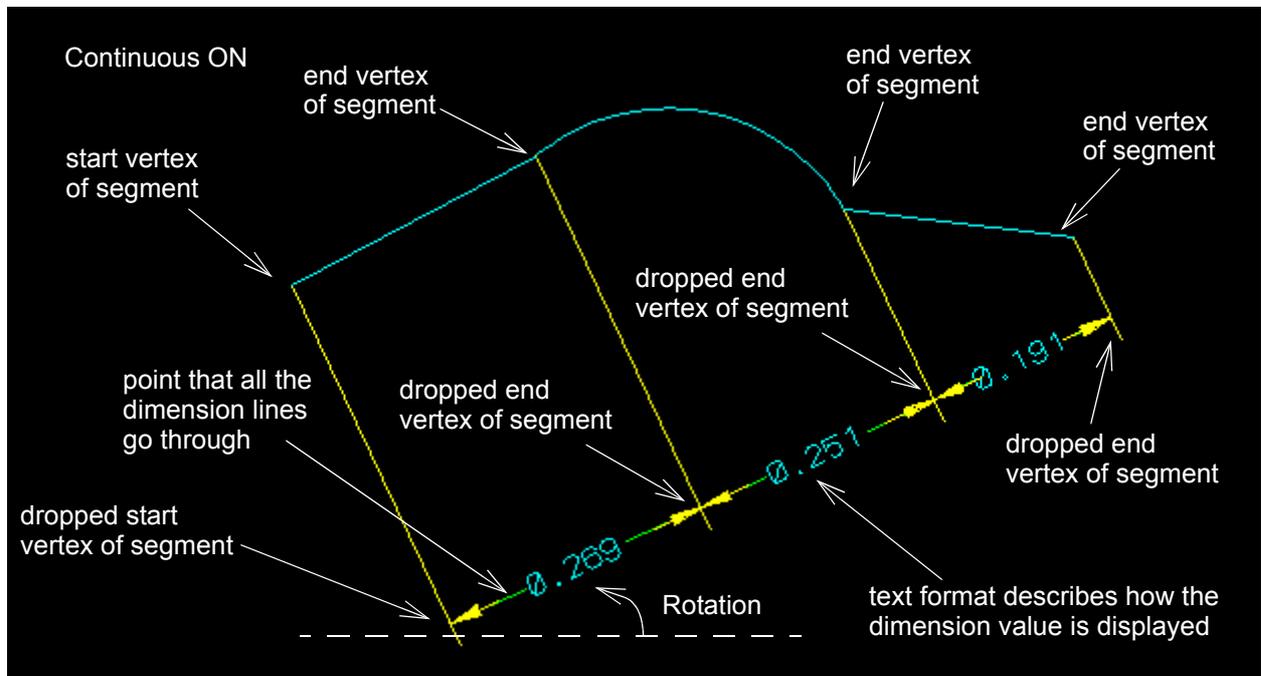
```
<[Pick segment] (a)ssociative (s)tyle[default] forma(t) [(c)ontinuous ON] an(g)le[10°] > [picks][fast][Menu]
```

When another segment is selected, the start and end vertices are dropped onto the line with the user given angle (Rotation) going through the current dimension point.

The **2D (plan) distance** is calculated between the dropped ends of the segment.

This repeats until Continuous is toggle **OFF** and then the next selected segment is the last **Rotated segment** dimension in the sequence.

The **Rotated segment** option then starts again. That is, goes back to **Step 1**.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#) .

## 18.1.7 Length

**Length** labels the **2D length** of a selected **segment**. The segment may be a straight, arc, transition or offset transition.

The dimension is added to the model given in the CAD toolbar.

**c** or **C** toggles Continuous **ON** and **OFF**. See [18.1.2 Continuous Mode](#).

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#).

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#).

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#).

**o** or **O** toggles offset **ON** and **OFF**. See [An Important Note about the Offset Toggle](#).

### Step 1. Pick the segment

Select **Length** and the following message is written to the screen message area

```
<[Pick segment] (a)ssociative (s)tyl[e[default] forma(t) [(o)ffset ON] [(c)ontinuous OFF] > [picks][fast][Menu]
```

If **association** is **ON** then the dimension is associated with the selected segment.

**Note:** Continuous mode is only available if **offset** is **ON**.

The 2D length of the selected segment is calculated.

### Step 2. Pick the dimension point

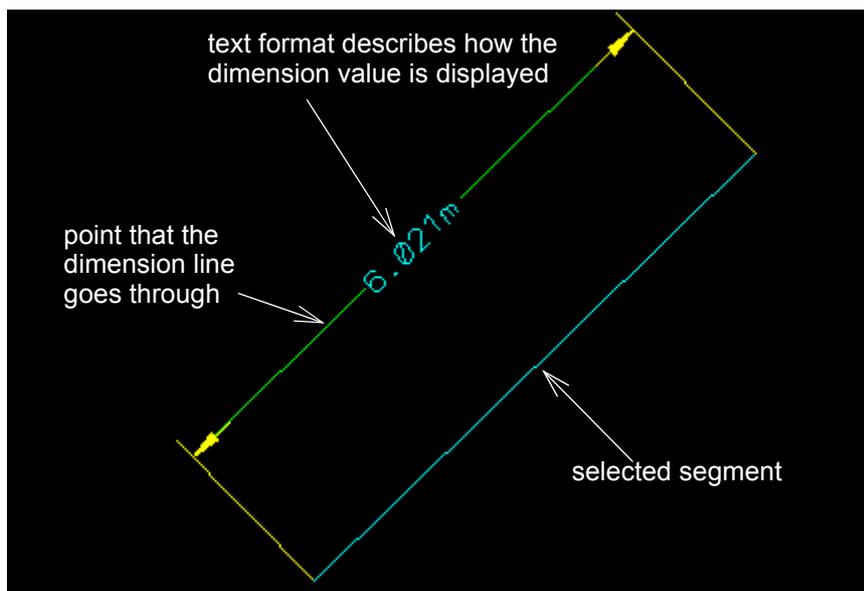
The following message is then written to the screen message area.

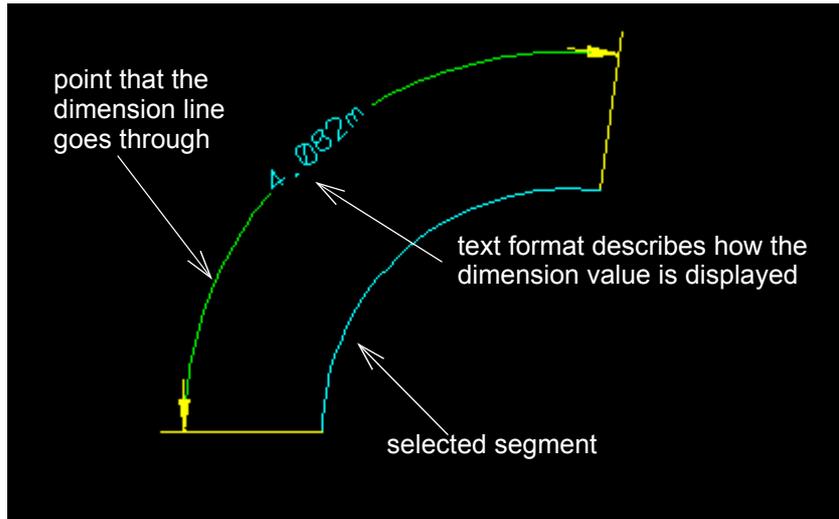
```
<Pick dimension point> [picks][fast][Menu]
```

The dimension line is then drawn parallel to the segment and going through the dimension point.

If **offset** is **ON** then the distance between the dimension point and the segment (the offset distance) is remembered with the dimension.

If **offset** is **OFF** then the dimension point itself is remembered with the dimension.





**Step 3.1**

If **Continuous** is **OFF**:

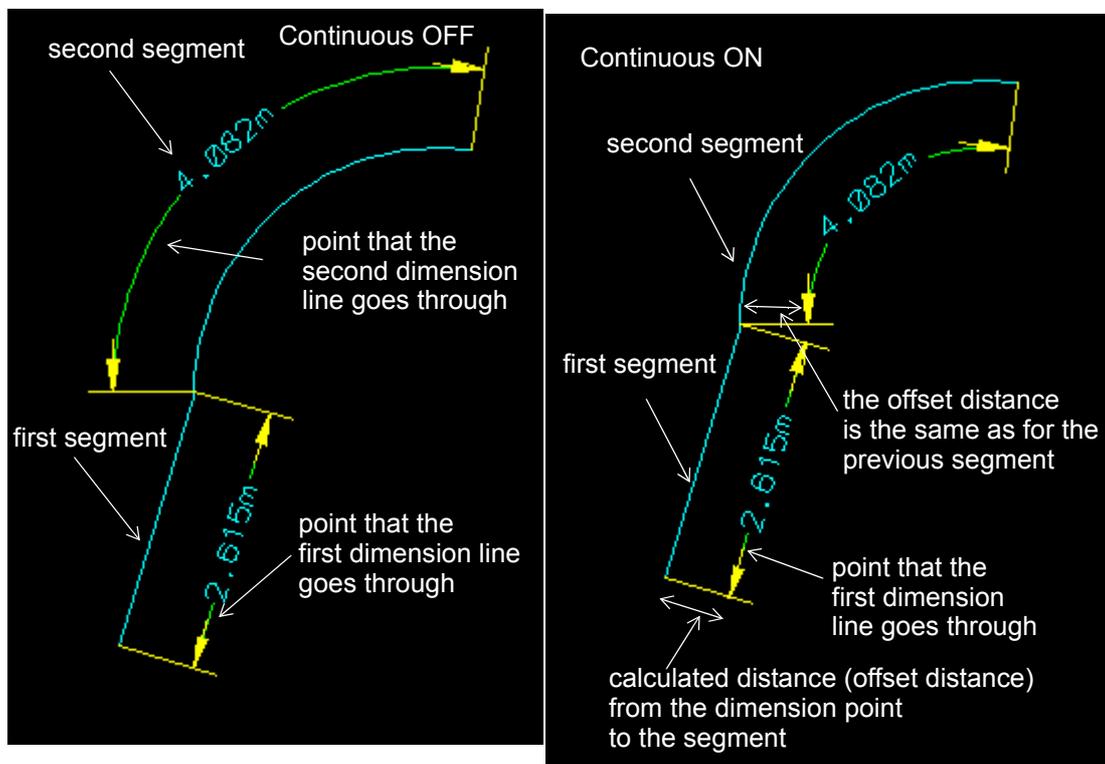
The **Length** option then starts again. That is, goes back to **Step 1**.

**Continuous** is **ON** (and for that **Offset** must also be ON):

If **Continuous** is **ON** then you are asked to pick another segment and another **Length** dimension is created for the next segment. The distance the dimension line is from the segment (**offset distance**) is the same as it was for the previous segment.

This repeats until **Continuous** is toggle **OFF** and then the next selected segment is the last **Length** dimension in the sequence.

The **Length** option then starts again. That is, goes back to **Step 1**.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

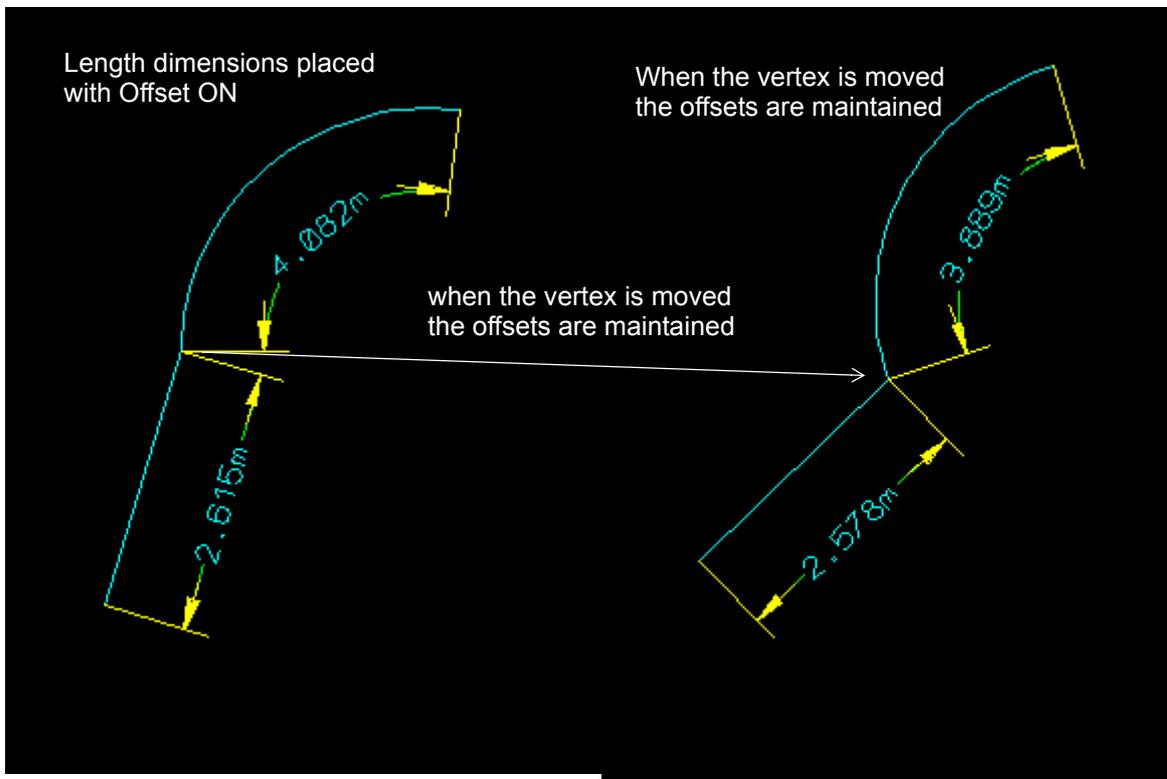
### An Important Note about the Offset Toggle

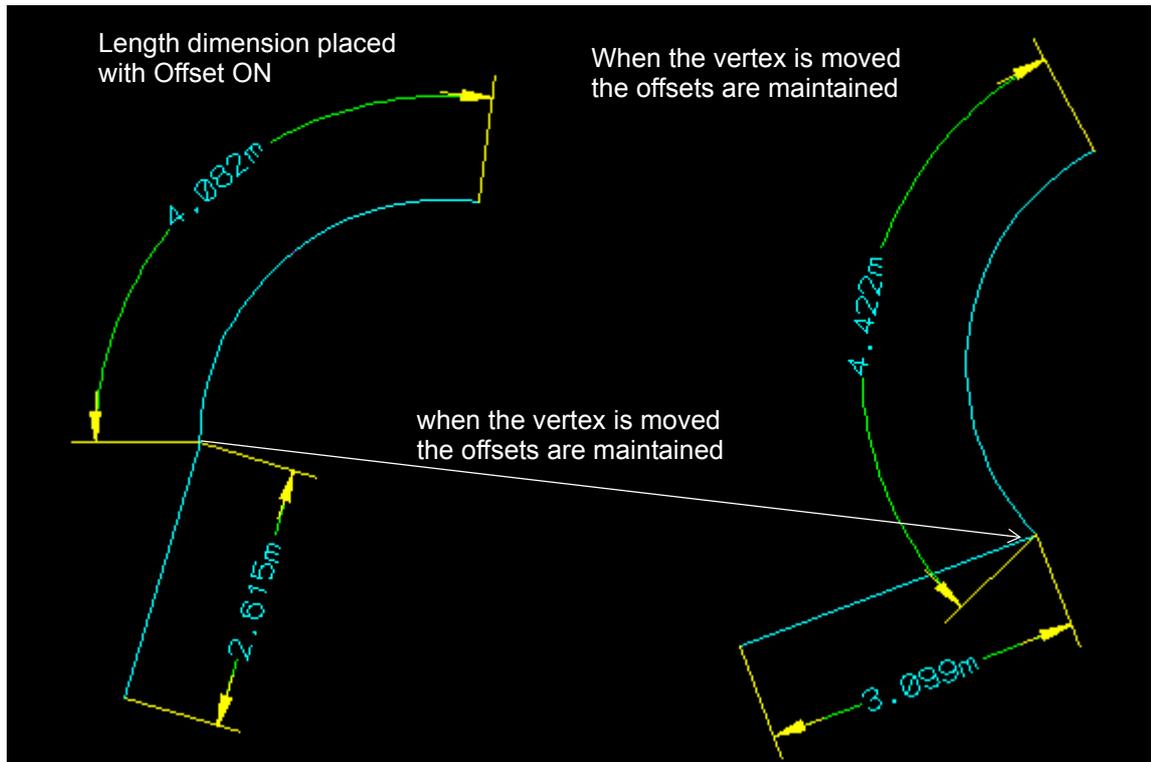
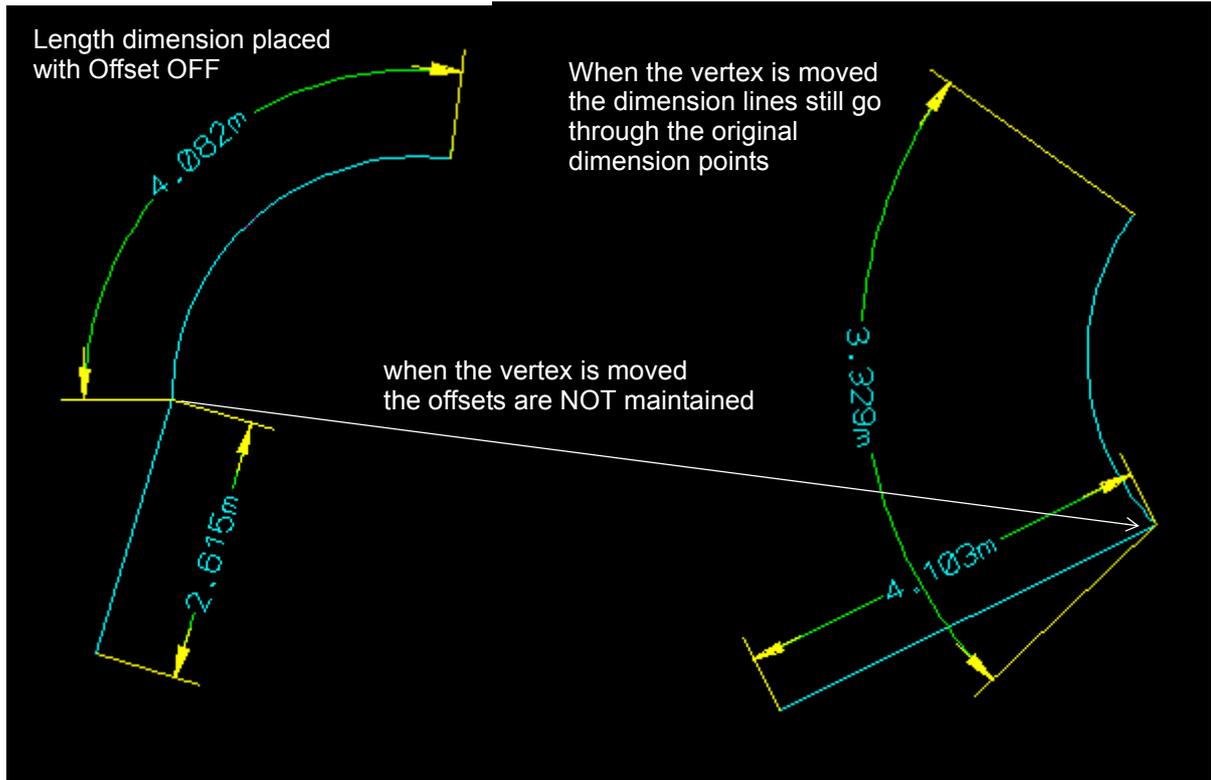
When the dimension line is first drawn, the difference between **Offset ON** and **Offset OFF** is not apparent.

But when you move a vertex and the associated **Length** dimensions go with it, the difference is move obvious.

For the **Length** dimensions that are created with **Offset ON**, the **original offsets** are **maintained**. So the new dimension lines will be the same parallel distance from the segments as when they were originally dimensioned.

For the **Length** dimensions that are created with **Offset OFF**, the **original dimensions points** are **maintained**. So the new dimension lines will go through the original dimension points and so their parallel distance from the segments will be totally different from before





## 18.1.8 Drop Segment

**Drop segment** labels the 2D distance between a selected point, and the point is dropped perpendicularly onto a selected base segment.

If the base segment is a line then the drop may occur on the extended segment.

If the base segment is an arc then it will drop to the outside or inside of the visible part of the arc.

If the base segment is a circle then the drop will go to the inside or outside of the circle depending on whether the point is inside or outside the circle.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

**z** or **Z** toggles *Zero Offset* mode **ON** and **OFF**. See [18.1.4 Zero Offset Mode](#)

### Step 1.

Select **Drop segment** and the following message is written to the screen message area

```
<[(a)ssociation ON] Pick start point or change format (t)ext (s)tyle[d1] [(z)ero offset ON] > [picks][fast][Menu]
```

#### Pick the start point.

If Association is **ON** then the start of the dimension is associated with the selected start point.

### Step 2.

The following message is written to the screen message area asking for a segment to be selected. The segment can be a straight, an arc, a transition or an offset transition.

```
<[(a)ssociation ON] Pick base segment> [picks][fast][Menu]
```

#### Pick the segment.

If Association is **ON** then the end of the dimension is associated with the selected segment.

The picked point is dropped perpendicularly onto the selected segment (or the extended line segment) and the 2D distance is calculated between these two points.

### Step 3.

If **Zero Offset** is **OFF**, the following message is then written to the screen message area.

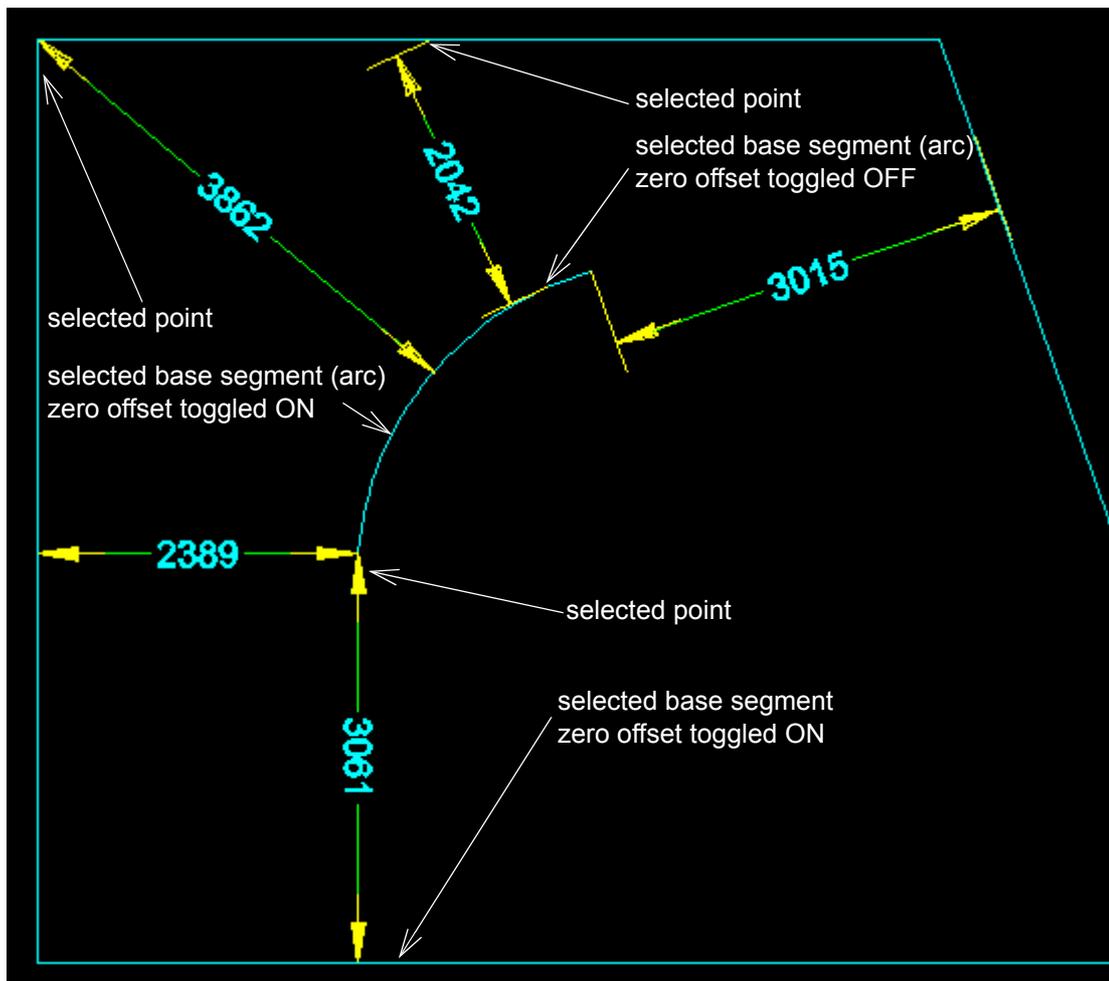
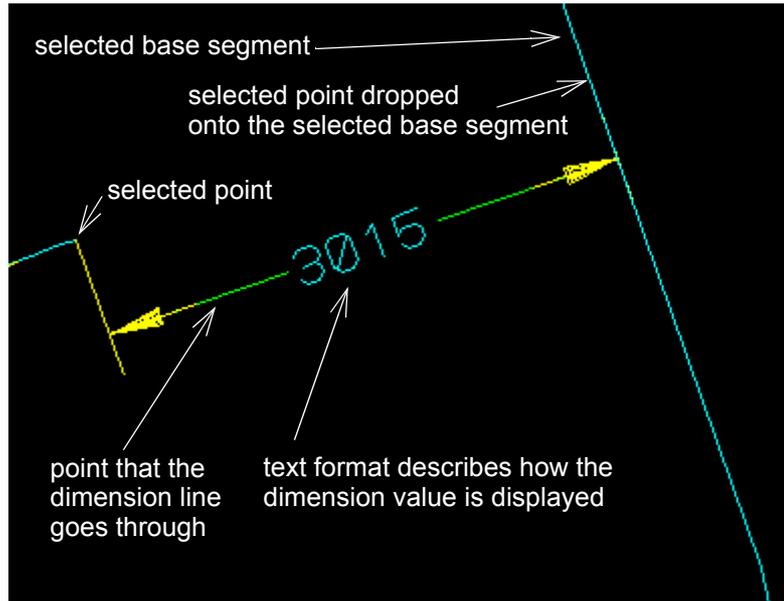
```
<Pick dimension point> [picks][fast][Menu]
```

Pick the point that the dimension line will go through (dimension point).

The dimension line is then drawn parallel to the line between the selected point and the selected point dropped perpendicularly onto the selected segment, and going through the dimension point.

If **Zero Offset** is **ON**, then no dimension point is picked because the dimension point is taken to be on the line joining the picked point and the dropped point.

For both cases of **Zero Offset**, the **Drop segment** option then starts again. That is, goes back to **Step 1**.



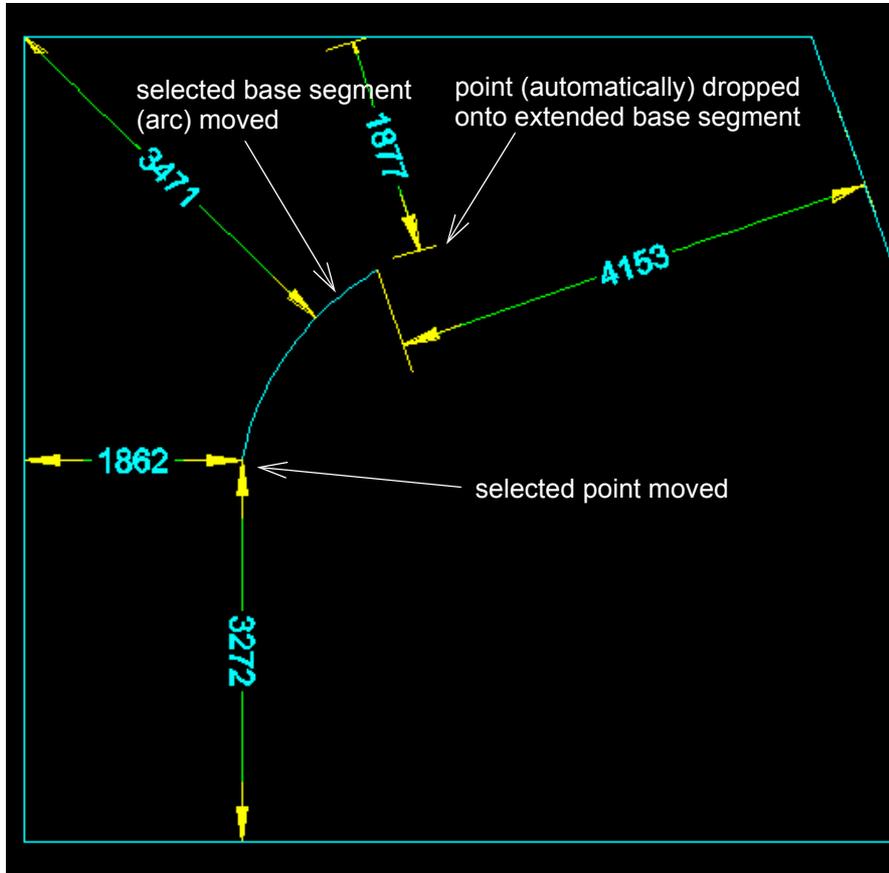
For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

**Note on Automatic Recalculation of Drop Segment Dimension When the Point or Segment Moves**

When **Association** is turned **ON**, the picked point and the picked base segment are associated

with the *Drop segment* dimension. So if either the picked point or the picked base segment is moved, the drop dimensions automatically recalcs.

Also note that the drop onto the base segment may involve the extended base segment because the drop point may not be between the end points of the selected base segment.



## 18.1.9 Drop String

**Drop string** labels the 2D distance between a selected point and the point dropped onto a selected base string.

By Dropped onto a string we mean that the point is first checked to see that it can be dropped perpendicularly onto a segment of the selected string (not an extended segment), or if there is no segment that the point can drop perpendicularly onto then it will drop onto the **nearest vertex** of the string.

If the base string is an arc then it will drop to the outside or inside of the visible part of the arc.

If the base string is a circle then the drop will go to the inside or outside of the circle depending on whether the point is inside or outside the circle.

Hence **Drop string** will always end on the base string.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#).

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#).

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#).

### Step 1. Pick the start point

Select **Drop string** and the following message is written to the screen message area

```
<[Pick start point] (a)ssociative (s)tyle[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected start point.

### Step 2. Pick the base string

The following message is written to the screen message area asking for a base string to be selected.

```
<(a)ssociative Pick base string> [picks][fast][Menu]
```

If Association is **ON** then the end of the dimension is associated with the selected base string.

The picked point is dropped onto the selected base string and the 2D distance is calculated between these two points.

The dimension line is then drawn between the selected point and the selected point dropped onto the selected base string.

The **Drop string** option then starts again. That is, goes back to **Step 1**.

## 18.1.10 Area

**Area** labels the plan area of a selected string.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

### Step 1. Pick the polygon to dimension

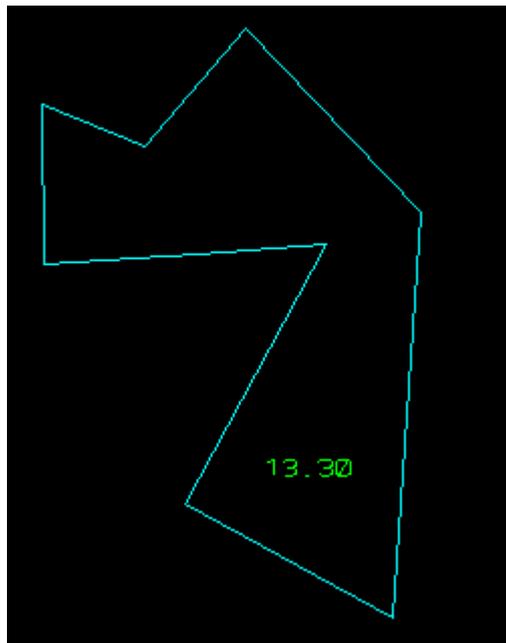
Select **Area** and the following message is written to the screen message area

```
<[Pick polygon] (a)ssociative (s)tyle[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the polygon.

If the selected string is not closed, the area is calculated for the polygon formed by joining the first and last vertices of the string.

The plan area is calculated for the polygon and placed inside the polygon.



The **Area** option then starts again. That is, goes back to **Step 1**.

For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

## 18.1.11 Jogged radius

This option has not yet been completed

- radius of a selected arc with jog

Pick an arc or a segment with a radius.

Then pick the start point of the jog and then the dimension point.

A dimension line is draw with a start symbol at the start and then a job between the dimension point and the start point of the jog.

The dimension is added to the model given in the CAD toolbar.

## 18.1.12 Diameter

This option has not yet been completed

**Diameter** labels the diameter of an arc or a segment with a non zero radius.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

### Step 1.

Select Diameter and the following message is written to the screen message area

```
<[(A)ssociation ON] Pick arc or choose (s)tyl[e] [diags - yes fixed len dim line] > [picks][fast][Menu]
```

**Pick the arc or segment with non zero radius.**

If Association is **ON** then the dimension is associated with the selected arc or segment.

The segment must have a radius. That is, it can't be a line segment.

The diameter is calculated for the arc or segment.

### Step 2.

The following message is then written to the left hand side of the screen message area.

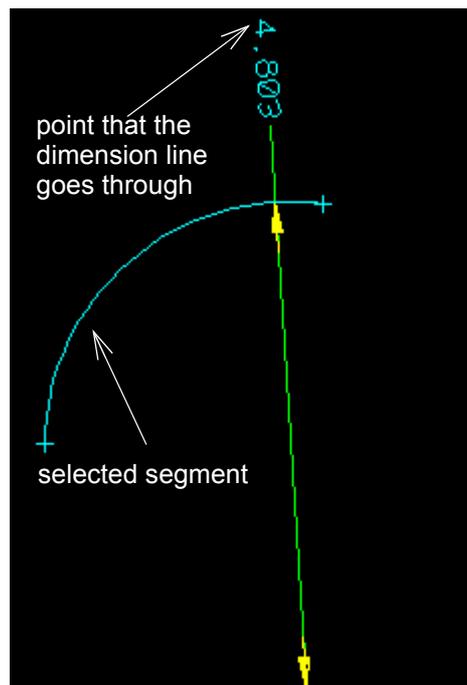
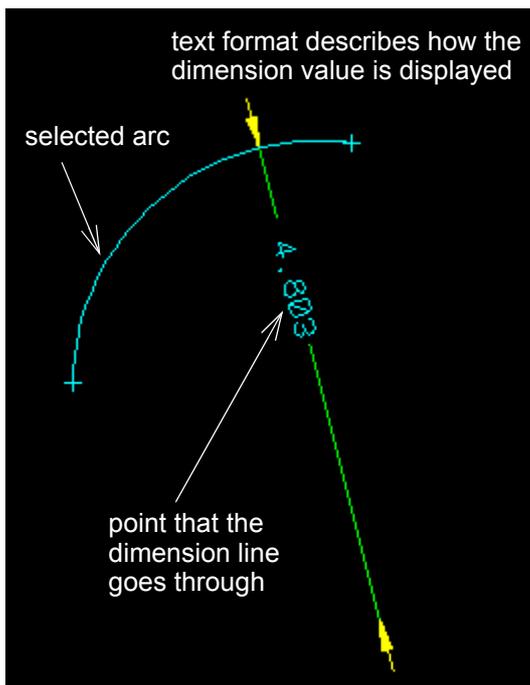
```
<Pick dimension point> [picks][fast][Menu]
```

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

**Pick the point that the dimension line will go through (dimension point).**

The dimension line is then drawn from the centre of the arc and through the dimension point and onto the arc. The text is placed at the dimension point.

If the dimension point is outside the arc, the dimension line continues through the arc.



The **Diameter** option then starts again. That is, goes back to Step 1.  
For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

## 18.1.13 Radius

This option has not yet been completed

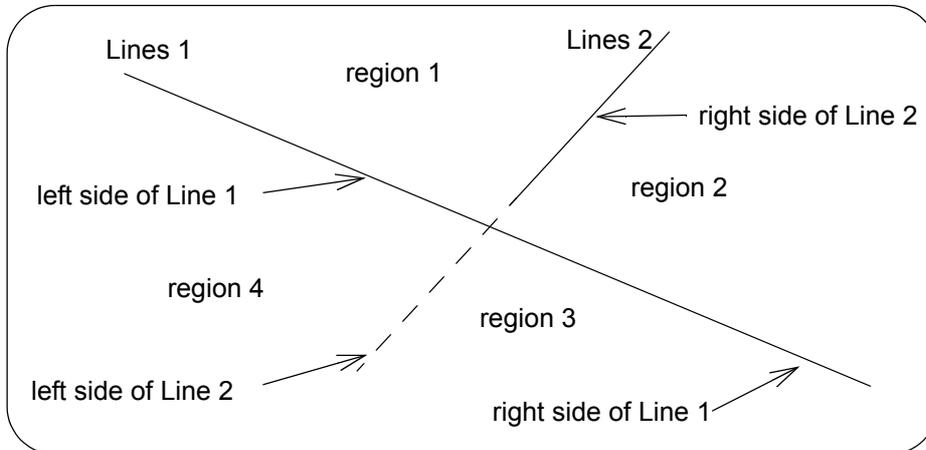
Radius of an arc or of a segment with a radius.

The dimension is added to the model given in the CAD toolbar.

## 18.1.14 Angle 2 Lines

**Angle 2 lines** labels the angle between two selected lines.

To define this dimension, the intersection point of the two extended lines divides the extended lines into two sides and the plane into four regions.



Although the lines can only be initially picked by the non extended part of the lines, the dimension can go between any of the halves of the extended lines.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#).

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#).

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#).

### Step 1. Pick the side to start the dimension from

Select **Angle 2 lines** and the following message is written to the screen message area

```
<[Pick start side] (a)ssociative (s)tyl[e] [default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected start line.

### Step 2. Pick the side to end the dimension on

The following message is written to the screen message area.

```
<[Pick end side] (a)ssociative > [picks][fast][Menu]
```

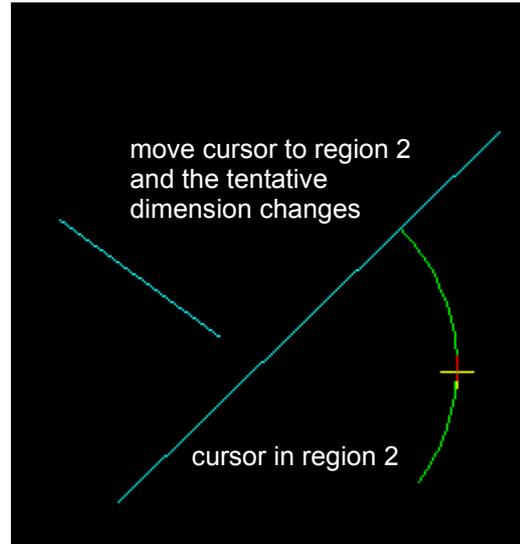
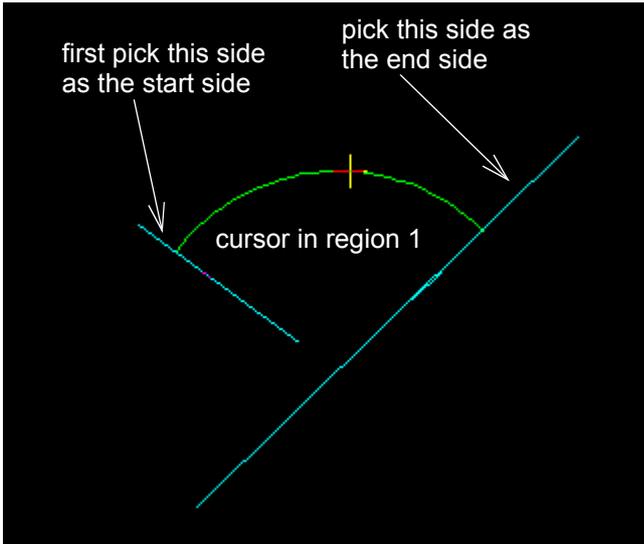
If Association is **ON** then the end of dimension is associated with the selected end line.

### Step 3.

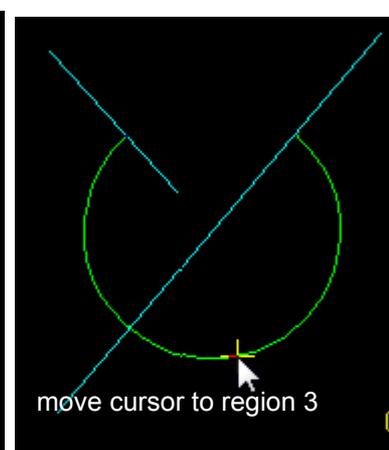
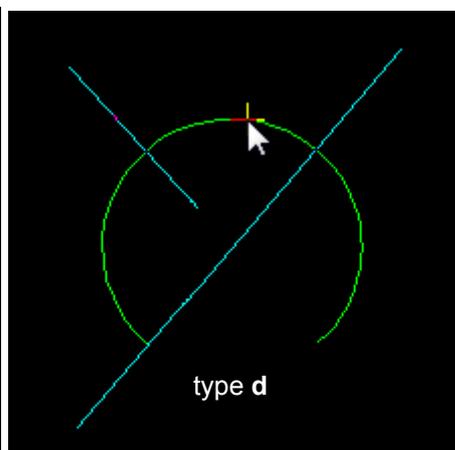
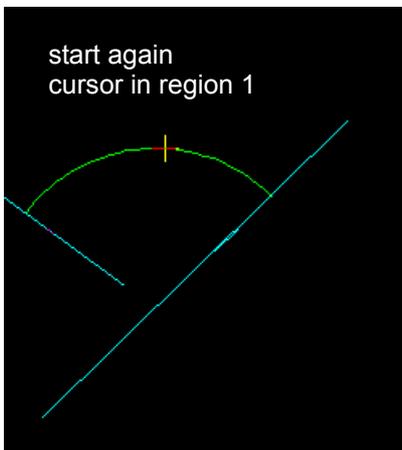
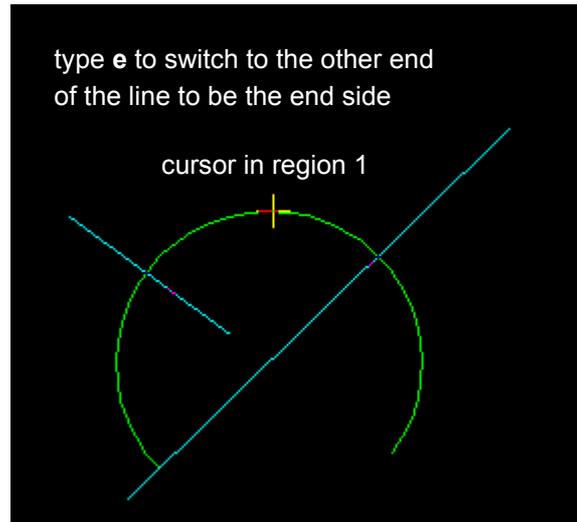
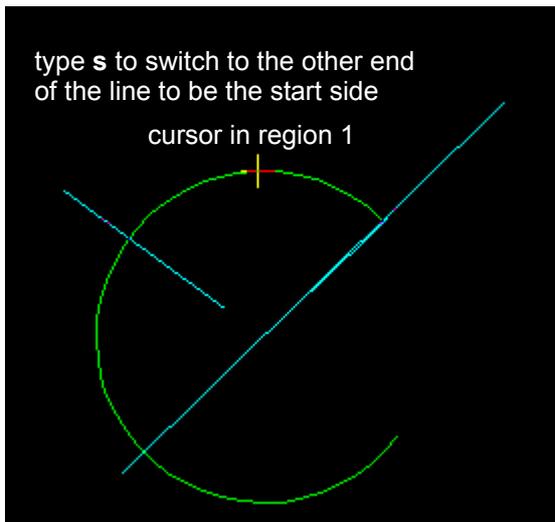
The following message is then written to the left hand side of the screen message area.

```
<Pick dimension point or reverse (d)irection (s)tart (e)nd line extension> [picks][fast][Menu]
```

As you move your cursor between the four regions, you will see the dimension line display as if the cursor was the selected dimension point.

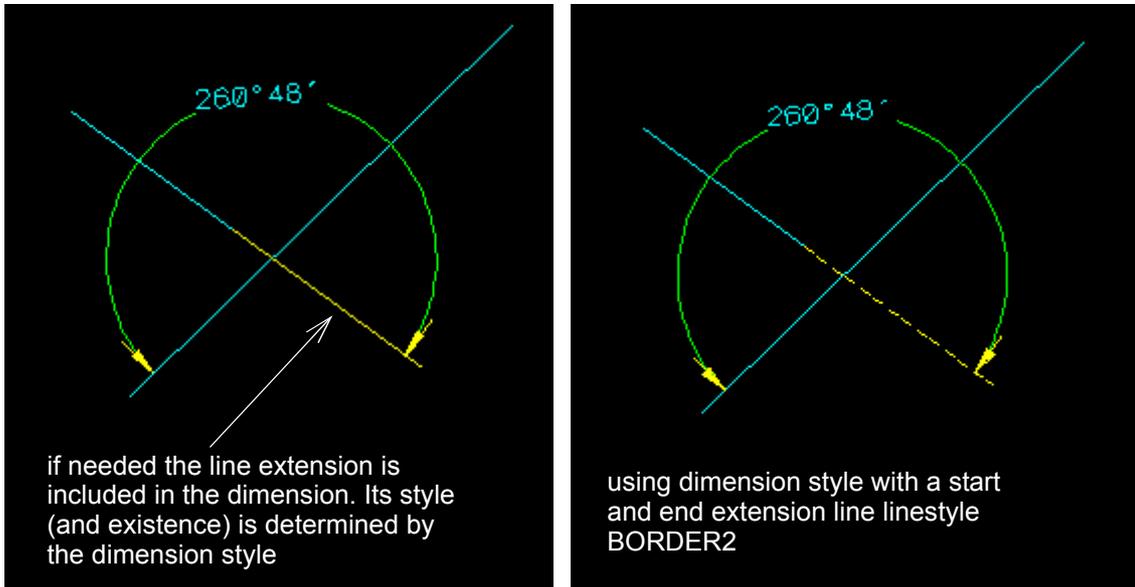


- s** or **S** takes the other end of the first line to start the dimension on.
- e** or **E** takes the other end of the second line to end the dimension on.
- d** or **D** toggles between a clockwise and counter clockwise angle for the dimension arc.
- t** or **T** can be typed to change the text format. See [18.3 Text Format for Dimensions and Leaders](#).



**Pick the point that the dimension line will go through (dimension point).**

The dimension line is then drawn from the start side to the end side and going through the dimension point.



The **Angle 2 lines** option then starts again. That is, goes back to **Step 1**.

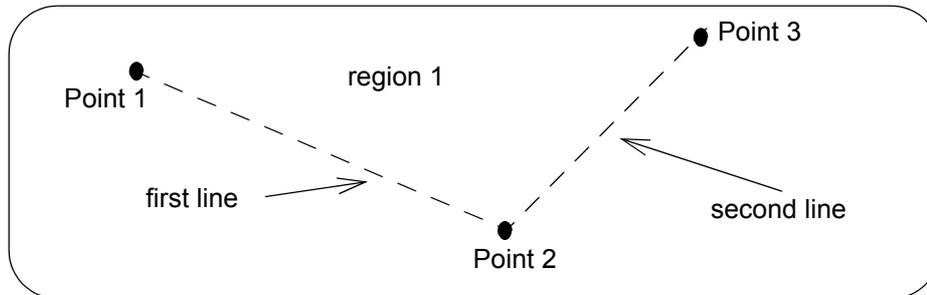
For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

## 18.1.15 Angle 3 points

**Angle 3 points** labels the angle between two temporary lines defined by 3 points. That is, it labels the angle between two lines that are constructed from the three points.

The first point is the start of the first line, the second point is the end of the first line and the start of the second line and the third point is the end of the second line.

The dimension is added to the model given in the CAD toolbar.



**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)  
**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)  
**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

### Step 1. Pick the first point of the first line

Select **Angle 3 points** and the following message is written to the screen message area

```
<[Pick start point] (a)ssociative (s)tyle[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the start of the dimension is associated with the selected point.

### Step 2. Pick the second point that is the end of the first line and the start of the second line

The following message is written to the screen message area.

```
<[Pick vertex point] (a)ssociative > [picks][fast][Menu]
```

If Association is **ON** then the vertex of the dimension is associated with the selected point.

### Step 3. Pick the third point that is the end of the second line

The following message is written to the screen message area.

```
<[Pick end point] (a)ssociative [(d)irection counter-clockwise] > [picks][fast][Menu]
```

If Association is **ON** then the end of the dimension is associated with the selected point.

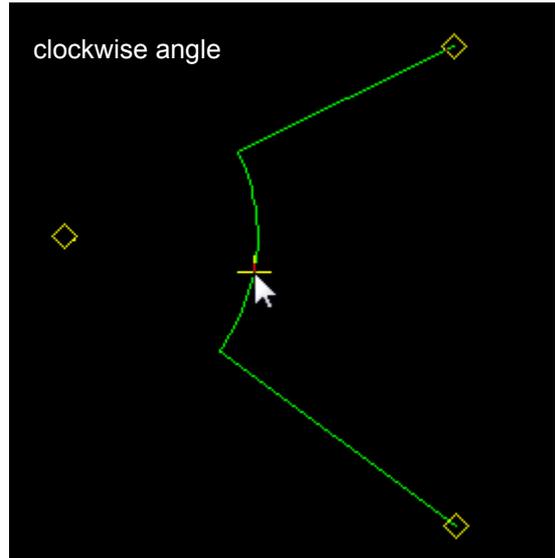
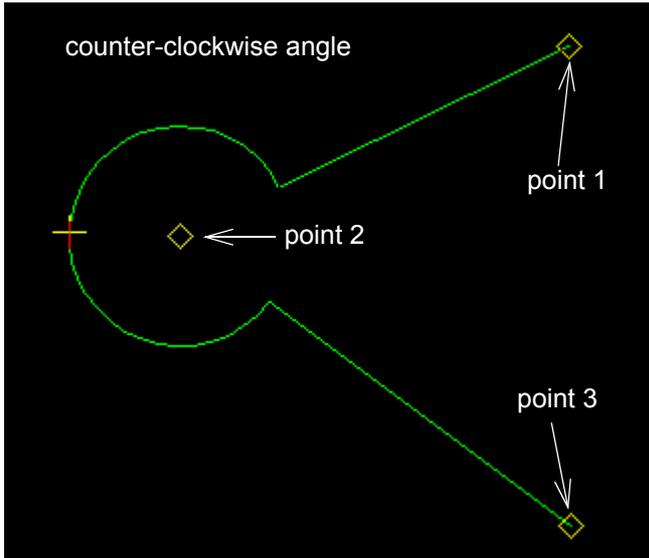
### Step 4. Pick the point that the dimension arc will go through (dimension point)

The following message is then written to the screen message area.

```
<[Pick dimension point] [(d)irection clockwise] > [picks][fast][Menu]
```

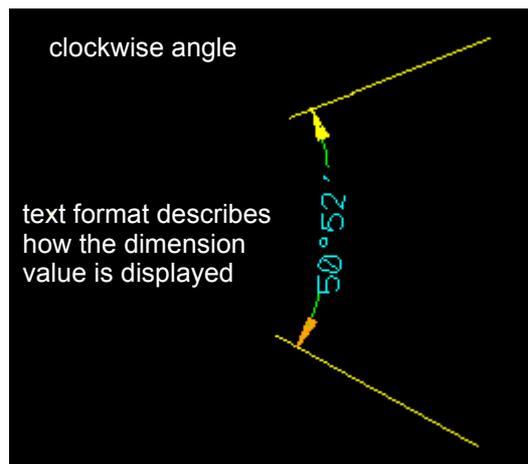
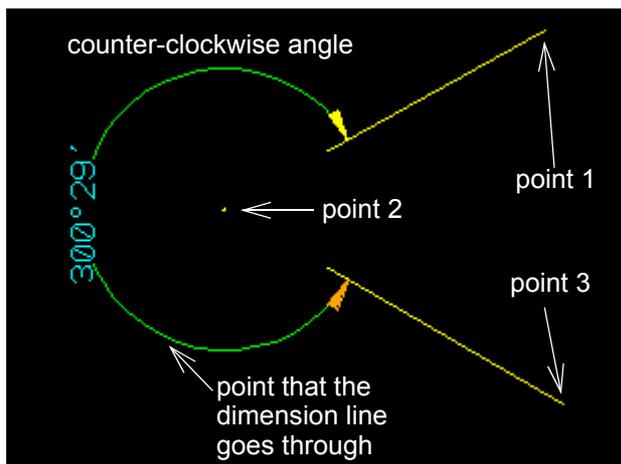
As you move your cursor you see how large the line of dimensions arc will be.

**d** or **D** toggles between a counter-clockwise and clockwise angle for the dimension arc.



When the dimension point is selected, the dimension arc is drawn from the start side to the end side and going through the dimension point.

The **Angle 3 points** option then starts again. That is, goes back to **Step 1**.



For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

## 18.1.16 Angle Arc

**Angle arc** labels the angle subtended by an existing arc, or of a segment with a radius.

The dimension is added to the model given in the CAD toolbar.

**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)

**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)

**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

### Step 1. Pick the arc of arc segment to dimension

Select **Angle arc** and the following message is written to the screen message area

```
<[Pick segment] (a)ssociative (s)tyle[default] forma(t) > [picks][fast][Menu]
```

If Association is **ON** then the dimension is associated with the polygon.

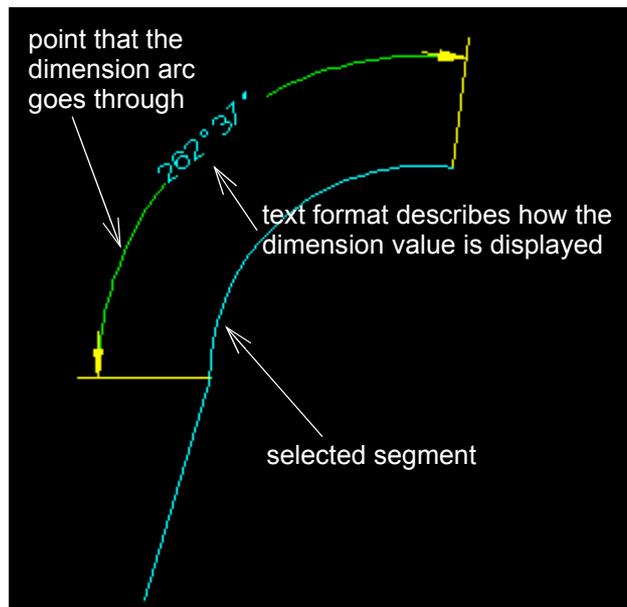
The angle of the arc can then be calculated.

### Step 2. Pick the point that the dimension arc will go through (dimension point)

The following message is then written to the left hand side of the screen message area.

```
<Pick dimension point> [picks][fast][Menu]
```

The dimension arc is then drawn parallel to the arc or segment and goes through the dimension point.



The **Angle arc** option then starts again. That is, goes back to **Step 1**.

For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

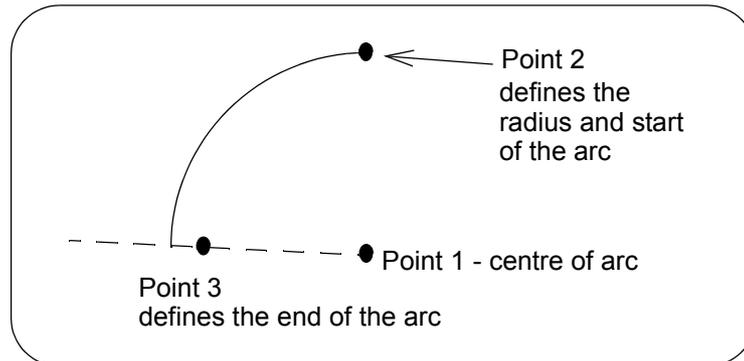
## 18.1.17 Arc Length by Centre, 2 Points

**NOT currently on the menu.**

**Arc** labels the length of an arc where the arc is temporarily defined by picking a centre point and two points on the arc.

Pick a position that will be centre point of the arc, and then the position that will be the first point of arc. This defines the centre, radius and start point of the arc.

A third position is then selected and it is dropped onto the arc defined by the first two points to give the end of the arc. This fully defines a temporary arc.



**a** or **A** toggles Association **ON** and **OFF**. See [18.1.1 Dimension Association](#)  
**s** or **S** brings up the dimension style list. [18.1.18.2 Dimension Styles](#)  
**t** or **T** brings up the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

### Step 1.

Select Arc and the following message is written to the screen message area

```
<[(a)ssociation ON] Pick centre point or choose (s)tyle[paper no clip orang end] c
```

#### Pick the position to be the centre point of the arc

If Association is **ON** then the start of the dimension is associated with the selected point.

### Step 2.

The following message is written to the screen message area.

```
<[(a)ssociation ON] Pick start point> [picks][fast][Menu]
```

#### Pick the second point is the start of the arc

If Association is **ON** then the vertex of the dimension is associated with the selected point.

### Step 3.

The following message is written to the screen message area.

```
<[(a)ssociation ON] Pick end point or change arc (d)irection [counter-clockwise] > [picks][fast][Menu]
```

**d** or **D** toggles between a counter-clockwise and clockwise angle for the dimension arc.

#### Pick the third point that is the end of the arc.

If Association is **ON** then the end of the dimension is associated with the selected point.

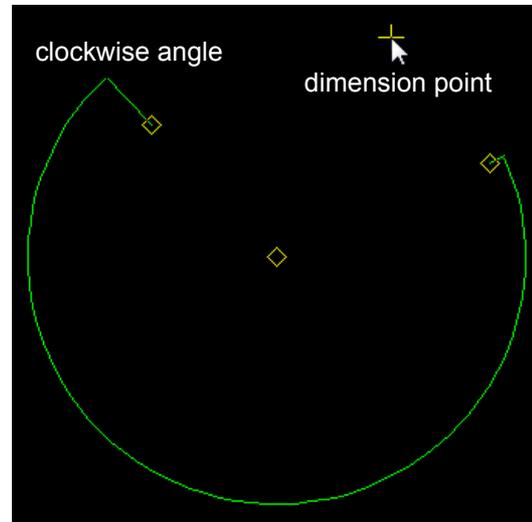
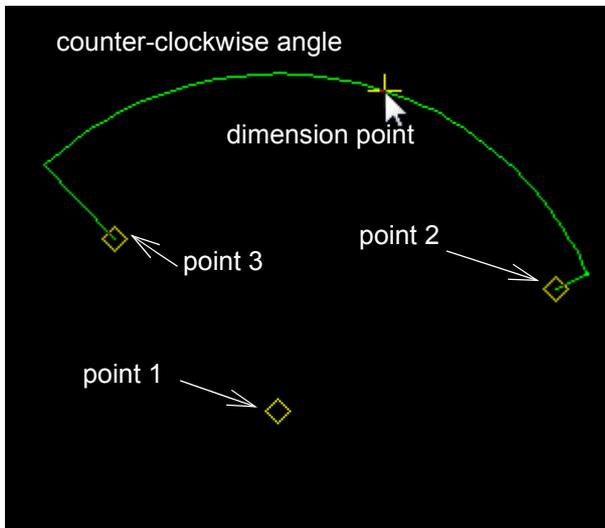
### Step 4.

The following message is then written to the left hand side of the screen message area.

```
<[Pick dimension point] [(d)irection clockwise] > [picks][fast][Menu]
```

As you move your cursor you see how large the line of dimensions arc will be.

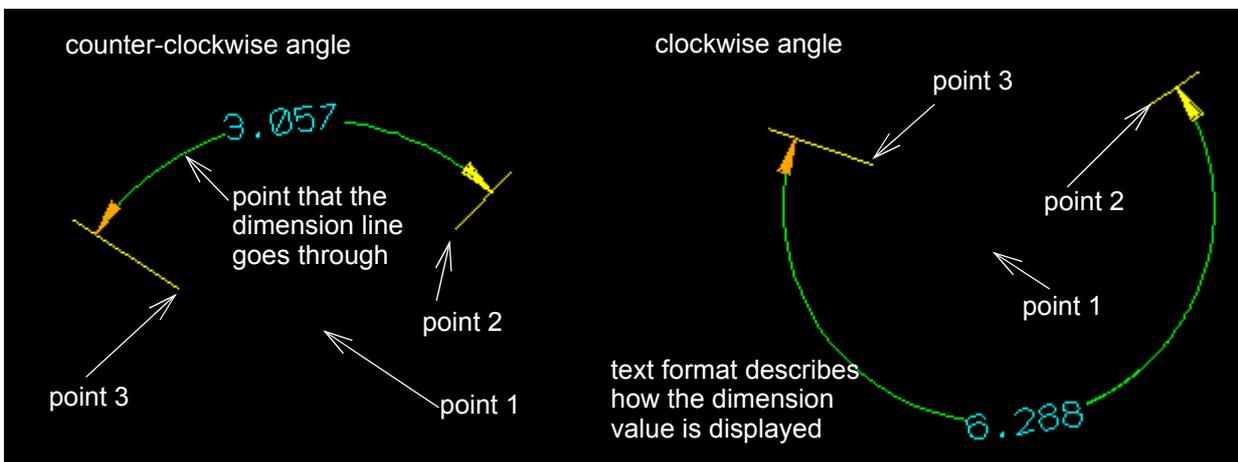
**d** or **D** toggles between a counter-clockwise and clockwise angle for the dimension arc.



**Pick the point that the dimension arc will go through (dimension point).**

The dimension arc is then drawn from the start side to the end side and going through the dimension point and labelled with the subtended angle.

The **Arc** option then starts again. That is, goes back to Step 1.

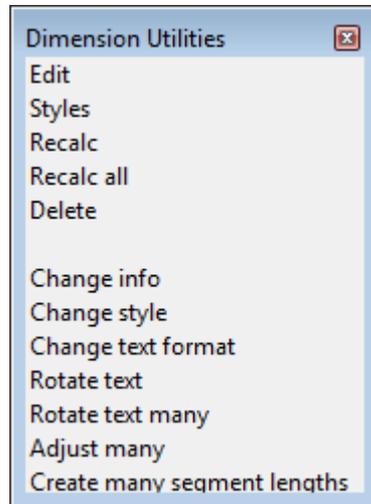


For information on how the dimension object appears, see [18.1.18.2 Dimension Styles](#).

## 18.1.18 Dimension Utilities

Position of menu: **Cad =>Dimension =>Utilities**

The **Dimension Utilities** menu is:



edit a dimension  
define styles for dimensions

See

[18.1.18.1 Dimension Edit](#)

[18.1.18.2 Dimension Styles](#)

**Recalc** - select a dimension to recalc.

**Recalc all** - recalcs all dimensions.

**Delete** - select a dimension to delete.

[18.1.18.3 Change Style of Dimension](#)

[18.1.18.4 Change Text Format](#)

[18.1.18.5 Create Many Segment Lengths](#)

### 18.1.18.1 Dimension Edit

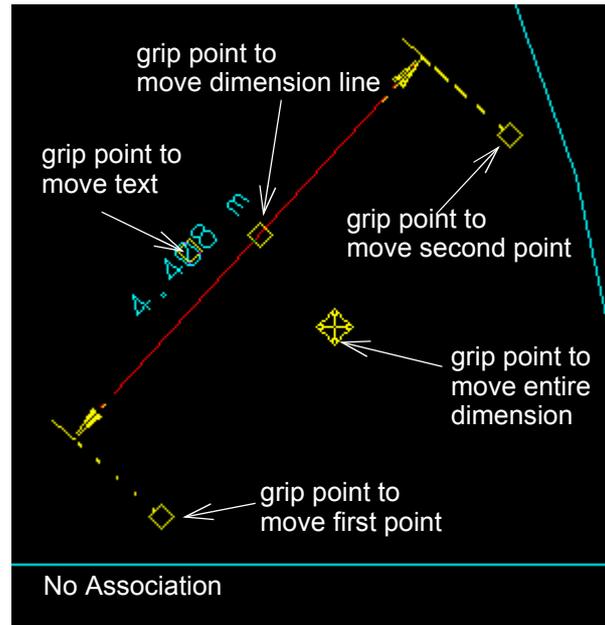
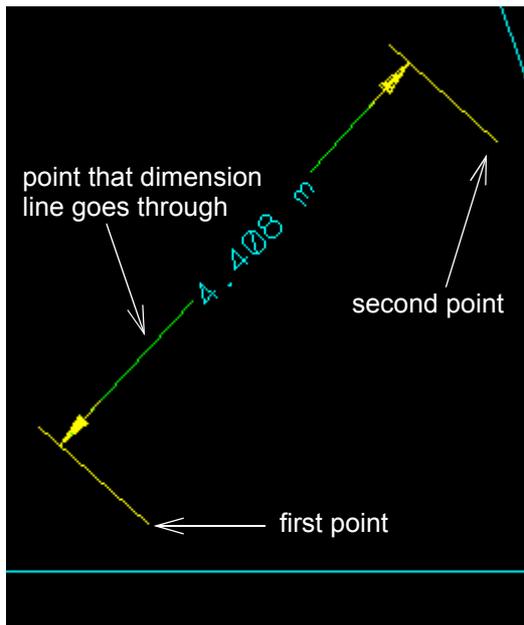
Picking an existing Dimension will bring up the Dimension editor.

In the Dimension editor, grip points relevant to the type of dimension will be displayed.

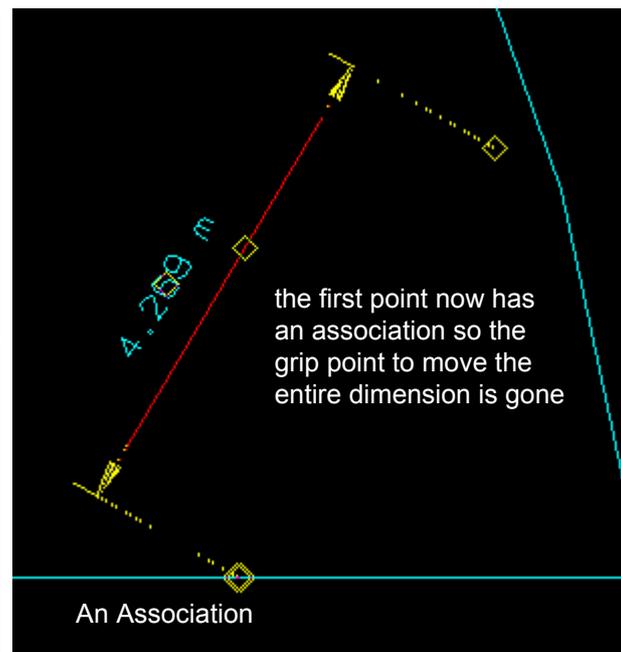
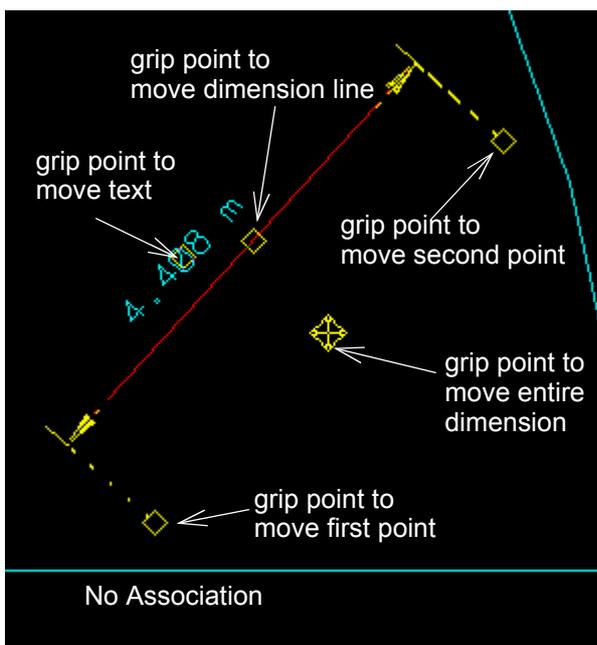
If there is **no Association** because either the Association is OFF, or Association is ON but there is nothing currently associated with the dimension, then you won't get a grip point to move the entire dimension because the dimension is locked to its associated items.

A list of typed options is also displayed in the screen message area.

```
<Pick grip point or change (t)ext or (o)blique angle or (s)tyl[e/diags] or (j)og or (d)elete or create (n)ew> [picks][fast][Menu]
```



Picking and accepting a grip point will move the grip point and when it is placed again, the association is recalced and the new value for the dimension displayed.



For the typed options, typing

**t** or **T** changes the dimension text format. See [18.3 Text Format for Dimensions and Leaders](#)

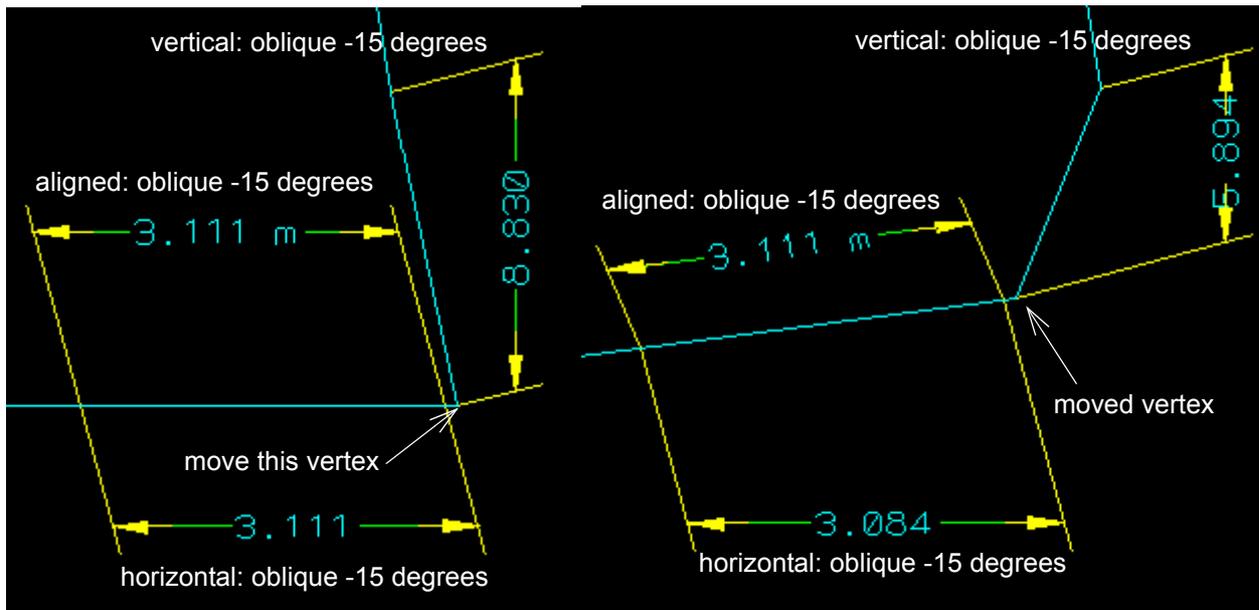
**o** or **O** changes the angle of the extension lines. The angle is measured in a counter clockwise direction and for horizontal dimensions it is measured from the positive y axis, vertical dimensions it is measured from the positive x axis and for aligned dimensions it is the measured from the perpendicular to the two points defining the dimension.

**s** or **S** changes the dimension style

**j** or **J** allows one **jog** to be inserted in a linear dimension

**d** or **D** deletes the dimension

**n** or **N** pick a new dimension to edit.



Also

Typing **<Esc>** will exit the editing of the current *dimension* and ask for a new drafting element (Leader or Dimension) to be selected for editing.

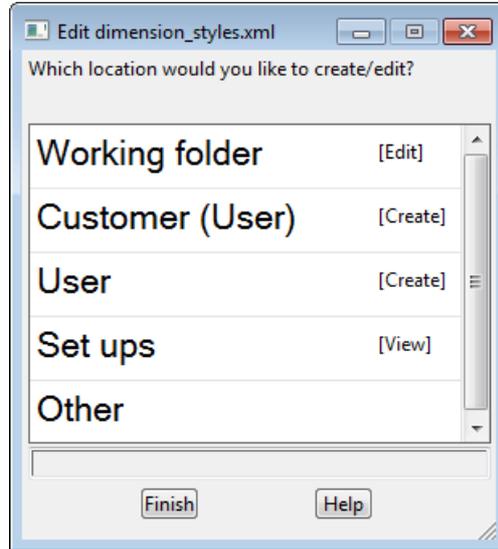
Similarly clicking **RB** and selecting **Cancel** from the **Pick Ops** menu will exit the editing of the current *dimension* and ask for a new drafting element (Leader or Dimension) to be selected for editing.

When back in the Drafting Editor, typing **<Esc>** or clicking **RB** and selecting **Cancel** from the **Pick Ops** menu, will exit the Drafting Editor

### 18.1.18.2 Dimension Styles

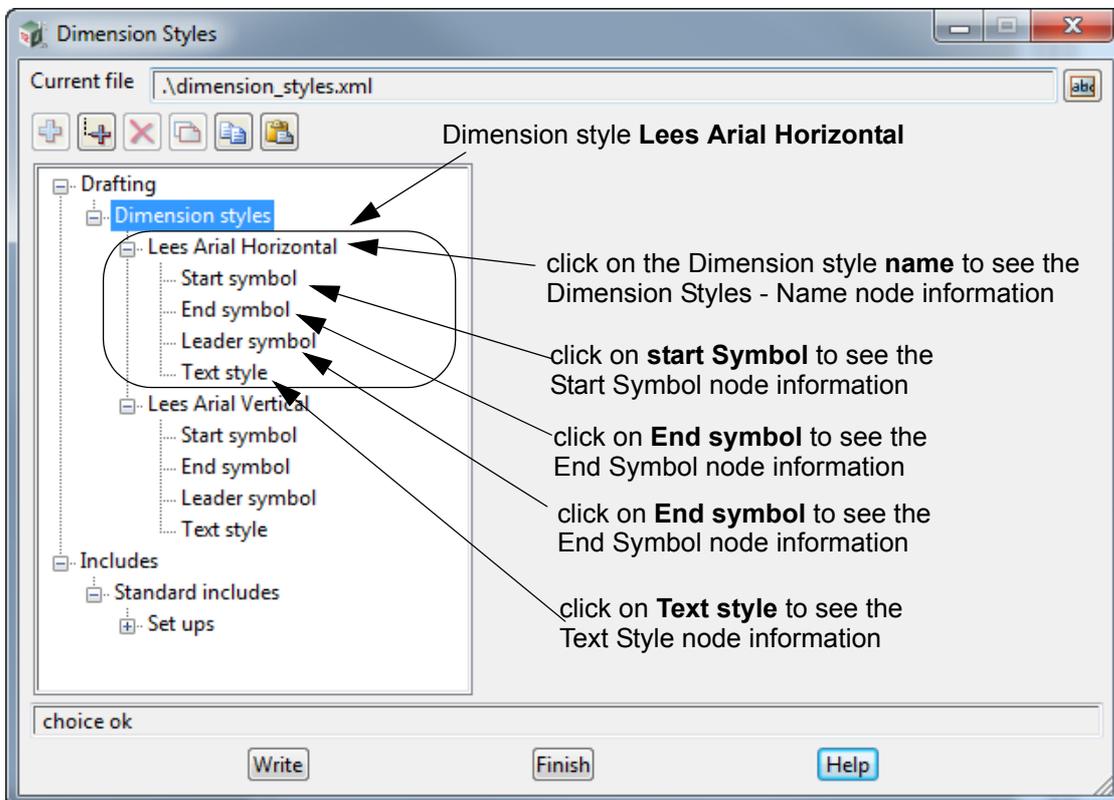
**Position of option on menu:** CAD =>Dimension =>Styles

When you click on the option an **Edit dimension\_styles.xml** panel is brought up and the panel shows the standard areas for looking for the table\_styles.XML files - Working folder, Customer (User), User, Set Ups and Other.



For information about where how to find and create dimension\_styles.xml files, see [18.5 Style XML Files for Dimensions, Leaders and Tables](#).

Once a dimension\_styles.xml file has been opened, the **Dimension Styles** panel is displayed.



See [18.1.18.2.1 Dimension Styles - Name Node](#)

[18.1.18.2.2 Dimension Styles - Start Symbol Node](#)

[18.1.18.2.3 Dimension Styles - End Symbol Node](#)

[18.1.18.2.4 Dimension Styles - Text style Node](#)

### 18.1.18.2.1 Dimension Styles - Name Node

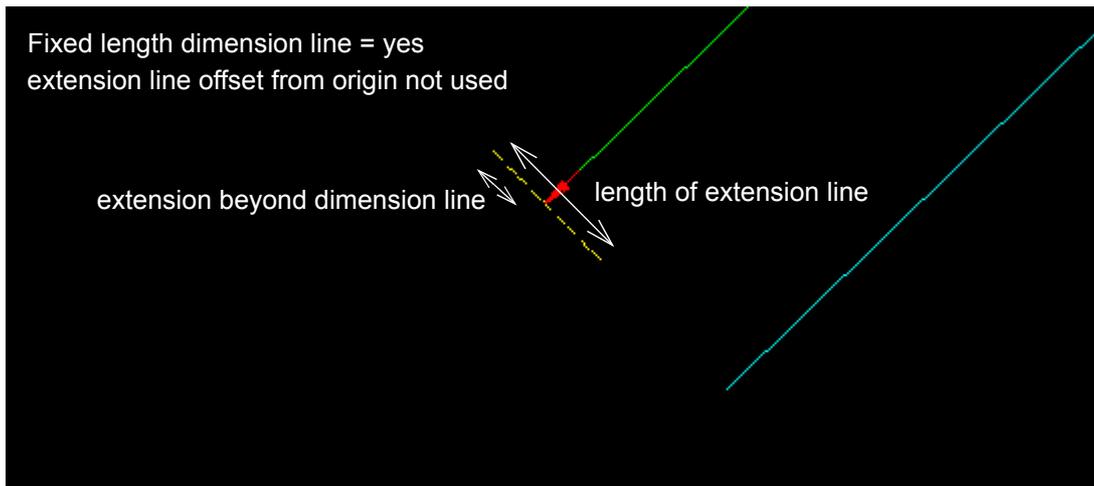
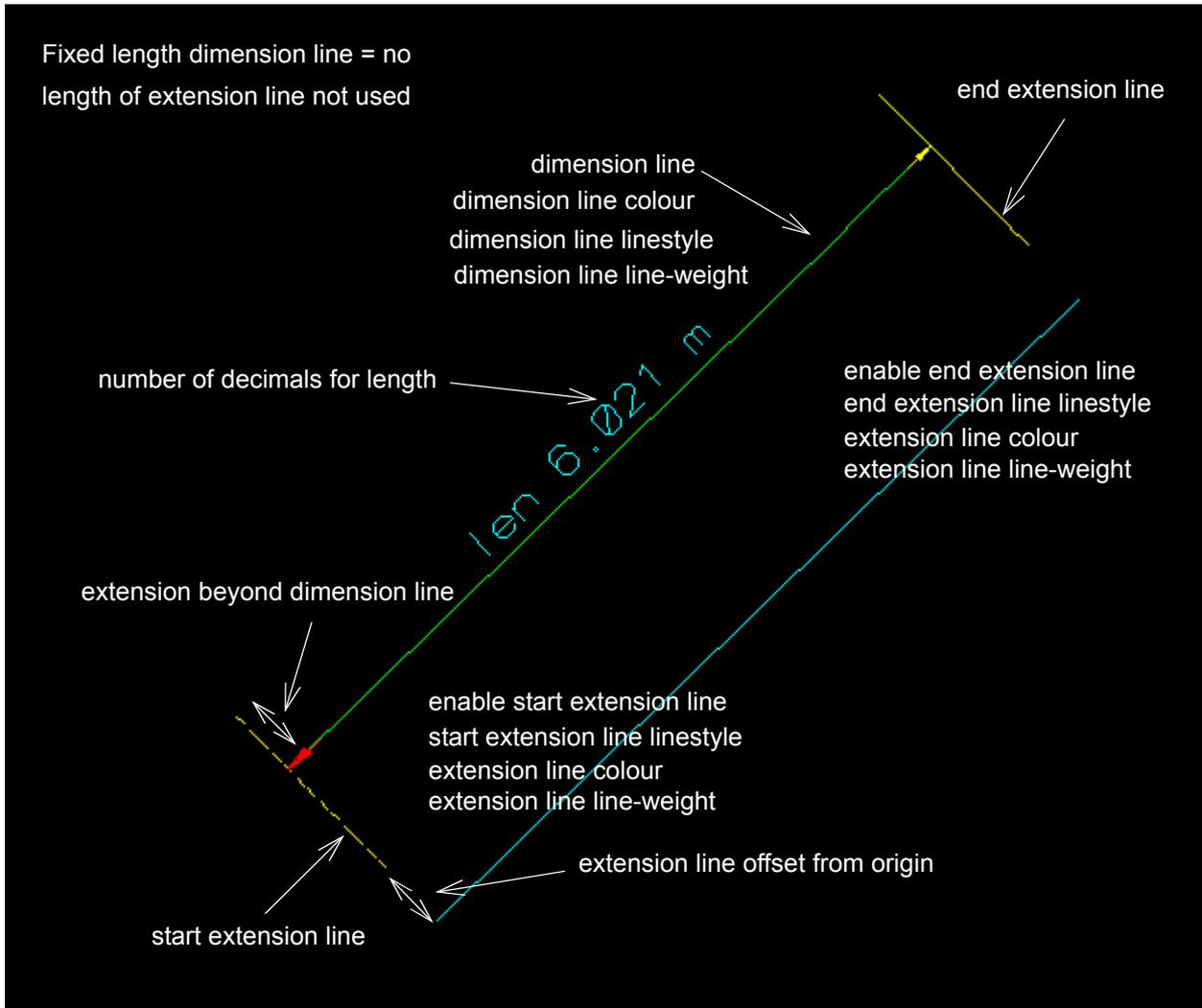
This is on the right hand side of the panel when you click on the *Dimension style* name.

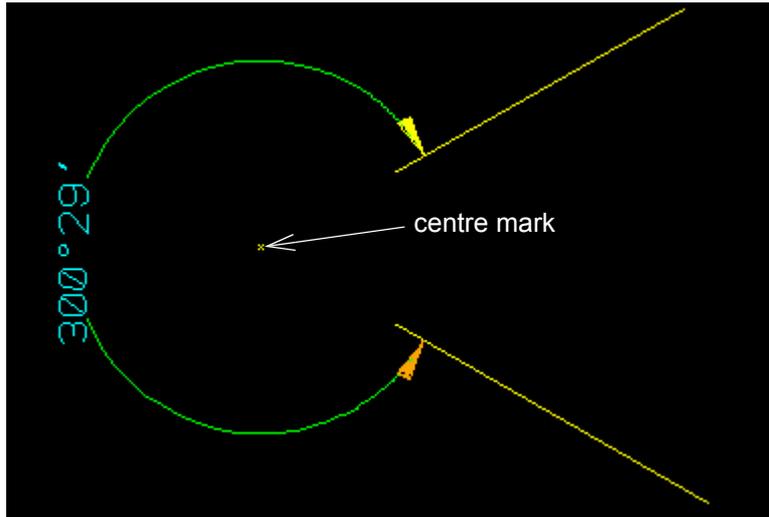
Name	d3	
Description	test	
Dimension line colour	green	
Dimension line linestyle	THICK	
Dimension line line-weight	0	
Extension line colour	blue	
Start extension line linestyle	BORDER	
End extension line linestyle	BORDER	
Extension line line-weight	0	
Enable start extension line	yes	
Enable end extension line	yes	
Extension beyond dimension line	0.2	
Extension line offset from origin	0.05	
Fixed length extension line	no	
Length of extension line	1	
Centre mark style	centre_mark	
Centre mark size	0.1	
Radius jog angle	0.888	
Radius jog factor	1.6	
Text alignment	text_aligned	
Forced text up	yes	
Angle for reading text	0	
Do not clip dimension line for text	no	
Use box around text	yes	
Use rounded box	yes	
Text box colour	blue	
Text box linestyle	THIN	
Number of decimals for length	3	
Number of decimals for area	2	
Number of decimal for angle	2	
Dimension line extension size	0	
Baseline spacing	1	
Enable start dimension line	yes	
Enable end dimension line	yes	
Dimension line break	0	
Length symbol style	length_preceding	
Text fit	fit_either	
Text offset	0	
Number of decimals for volume	2	
Angle format style	show_angle	

only used with some dimensions

if Do not clip dimension line for text is **yes** or Textstyle Type is not world then these are not used  
if Do not clip dimension line for text is **no** and Textstyle Type is world, then these are used

these settings are not currently used





If **Don't clip dimension line for text** is set to **no**, or the **Type** from the **Textstyle** tab is not **world**, then the following are not used.

- Use box around text
- Use rounded box
- Text box colour
- Text box linestyle

The following items that are at the bottom of the tab have not yet been implemented.

- Dimension line extension size
- Baseline spacing
- Enable start dimension line
- Enable end dimension line
- Dimension line break
- Length symbol style
- Text fit
- Text offset
- Number of decimals for volumes
- Angle format style

### 18.1.18.2.2 Dimension Styles - Start Symbol Node

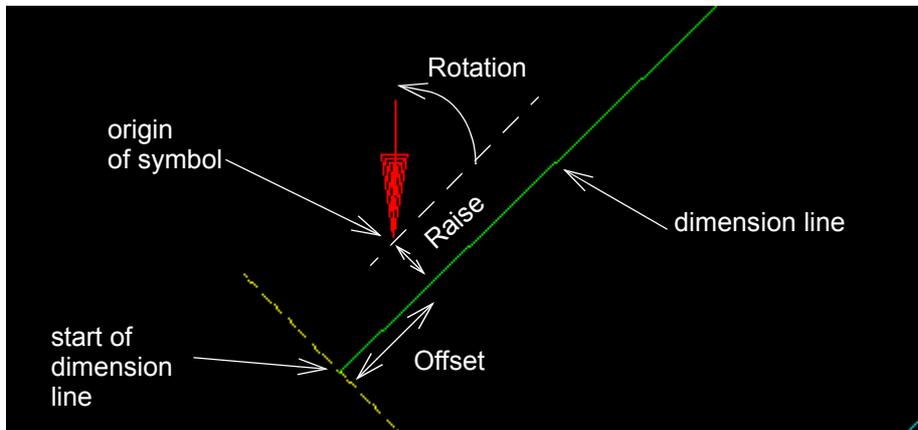
This is on the right hand side of the panel when you click on the **Start symbol** node.

Style	Arrow	
Colour	red	
Size	0.5	
Rotation	45	
Offset	0.2	
Raise	0.5	

For the **Start symbol**, the origin of the symbol is displaced from the **start** of the dimension line by the **Offset** and **Raise** values.

With a **Rotation** value of zero, the symbol is given the same rotation as the dimension line.

For a non zero **Rotation**, the value is **added** to the angle of the dimension line.



### 18.1.18.2.3 Dimension Styles - End Symbol Node

This is on the right hand side of the panel when you click on the **End symbol** node.

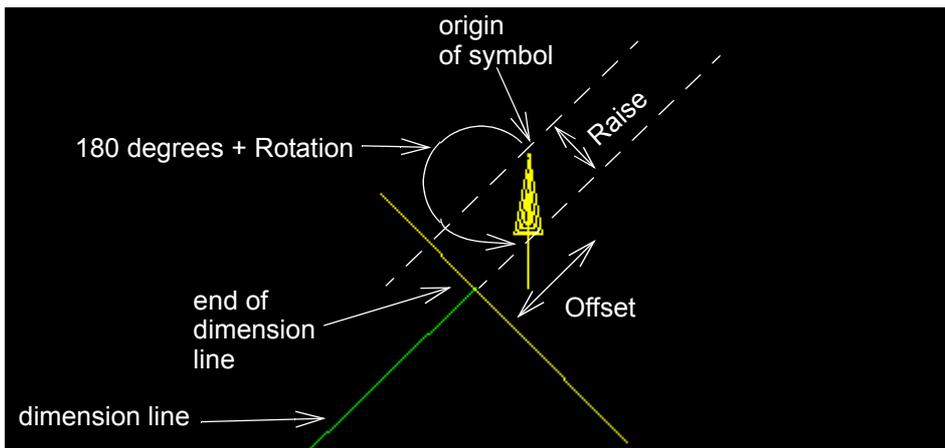
Style	ARROW	
Colour	yellow	
Size	0.5	
Rotation	45	
Offset	0.2	
Raise	0.5	

For the **End symbol**, the origin of the symbol is displaced from the **end** of the dimension line by the **Offset** and **Raise** values.

With a **Rotation** value of zero, the symbol is given the angle of the dimension line **plus 180 degrees**.

For a non zero **Rotation**, the value **plus 180 degrees** is **added** to the angle of the dimension line.

The addition of 180 degrees for the angle of the end symbol is so that the same symbol can be used for the start and the end of the dimension line.



### 18.1.18.2.4 Dimension Styles - Text style Node

This is on the right hand side of the panel when you click on the **Text style** node.

Textstyle	Arial	T
Name		abd
Type	paper	▼
Size	0.2	TF
Angle	0°	TF
X-factor	1	TF
Slant	0	TF
Offset	0	TF
Raise	0	TF
Colour	cyan	■
Whiteout	no colour	
Border	no colour	
Border type		▼
Justify	bottom-left	▼
TTF Underline	no	▼
TTF Strikeout	no	▼
TTF Italic	no	▼
TTF Outline	no	▼
TTF Weight	400	TF

This is expected to be modified shortly so that it uses a *Textstyle Data* and all the features from *Textstyle Data* are automatically available. For the moment with paper text, only **Whiteout** is supported.

When the *Textstyle Data* is implemented, it can have its own *Border type* but there is no linestyle for the border in *Textstyle Data*. The only way to get a border with a linestyle is to use World text and set the border parameters in the *General* tab ([18.1.18.2.1 Dimension Styles - Name Node](#)).

If **Type** is not *world*, then **Don't clip dimension line for text** is set to **no** and these items from the *General* tab are not used.

- Use box around text
- Use rounded box
- Text box colour
- Text box linestyle

See [18.1.18.2.1 Dimension Styles - Name Node](#)

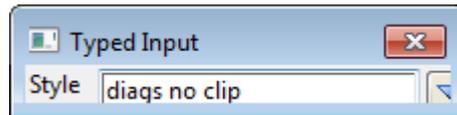
### 18.1.18.3 Change Style of Dimension

Click on **Change style of dimension** and the following message is written to the screen message area

```
<Pick drafting element to apply (s)tyle[diags no clip]> [picks][fast][Menu]
```

Pick a dimension and its style will be changed to that shown in the square brackets ([ ]). Then pick another dimension to change its style.

Typing **s** or **S** brings up the **Style Typed Input** box to select a new dimension style.



The new style is picked from the pop up list, or typed into the box and <Enter> pressed.

The box is then removed and the message again written to the screen message area.

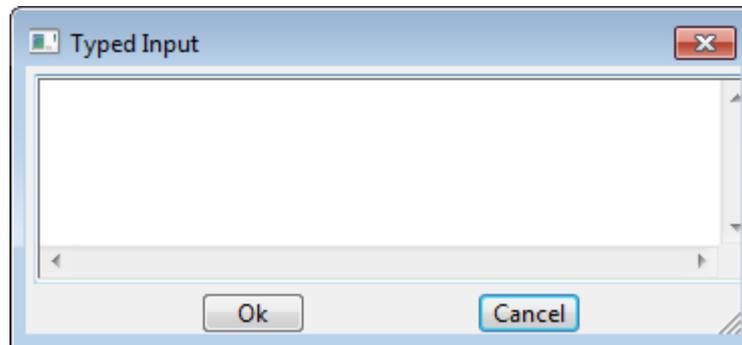
All subsequent selected dimensions will have their styles changed to this new style until either the option is terminated or a new dimension style is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

#### 18.1.18.4 Change Text Format

The **Change text format** option is used to change the text format for **Dimensions** or **Leaders**.

Click on **Change text format** and a **Typed Input** box comes up



and whilst you are in the **Typed Input** box the following message is written to the screen message area.

Enter Text [Caret][Same as][Menu] select a button

The new text format is typed into the **Typed Input** box and **OK** clicked.

The following message is written to the screen message area

<Pick drafting element to apply forma(t)[> [picks][fast][Menu]

Pick a **Dimension** or **Leader** and its text format will be changed to that shown in the square brackets ([ ]). Then pick another **Dimension** or **Leader** to change its text format

Typing **t** or **T** will bring the **Typed Input** box to type in a new text format. See [18.3 Text Format for Dimensions and Leaders](#)

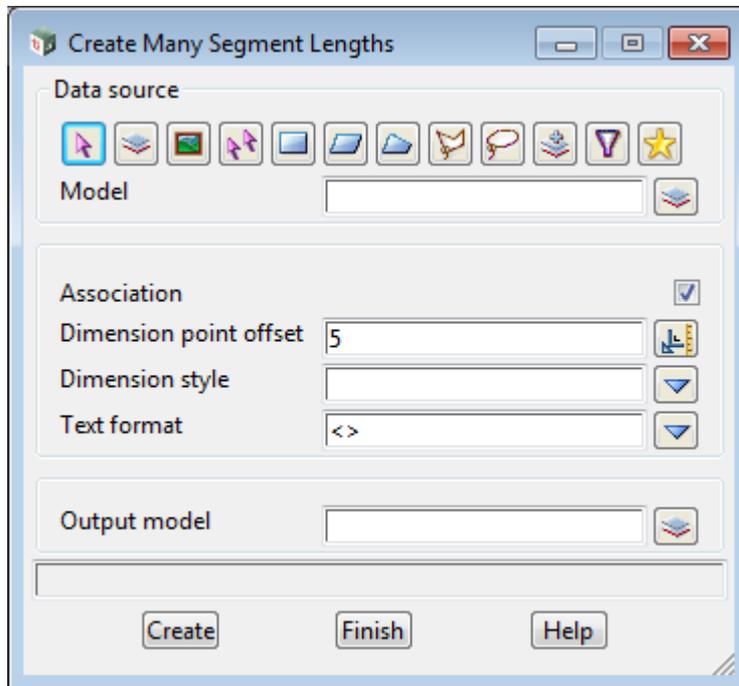
All subsequent selected **Dimensions** or **Leaders** will have their text formats changed to this new text format until either the option is terminated or a new text format is selected

To terminate the option, press **<Esc>**, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

### 18.1.18.5 Create Many Segment Lengths

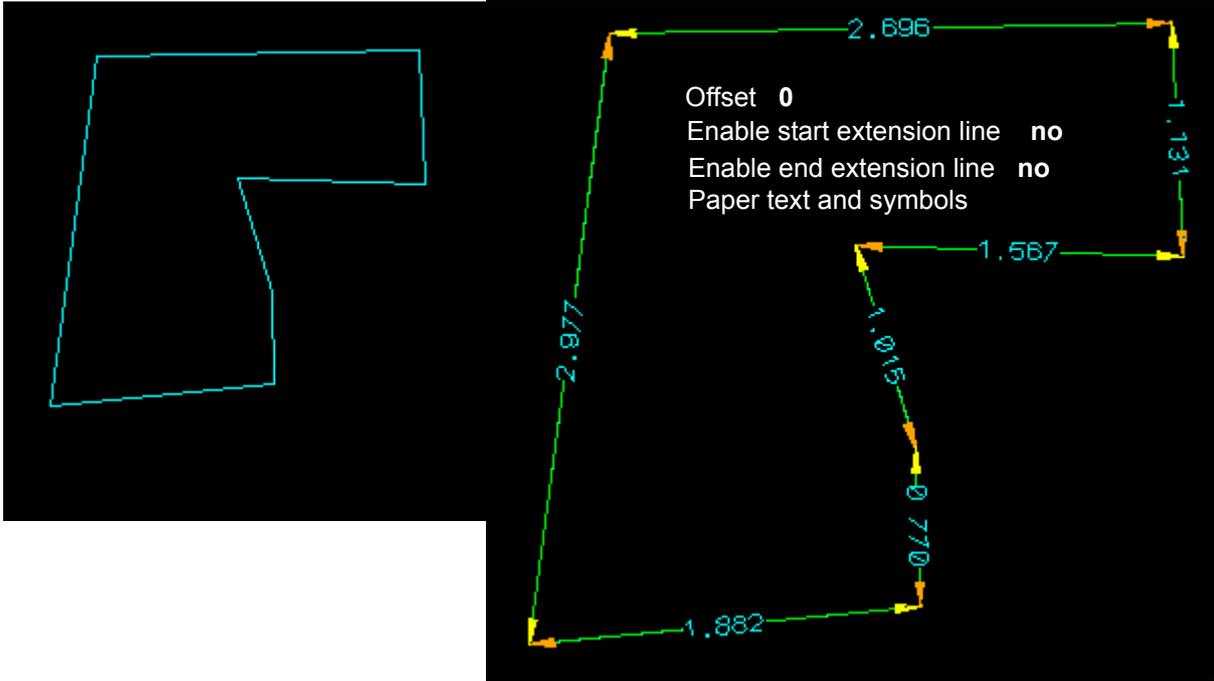
This option creates Length dimensions for all the segments of the strings in the Data Source. The created **Length** Dimensions all have the given **Offset** and **Text format**.

Selecting the **Create many segment lengths** brings up the **Create Many Segment Lengths** panel.



The fields and buttons used in this panel have the following functions.

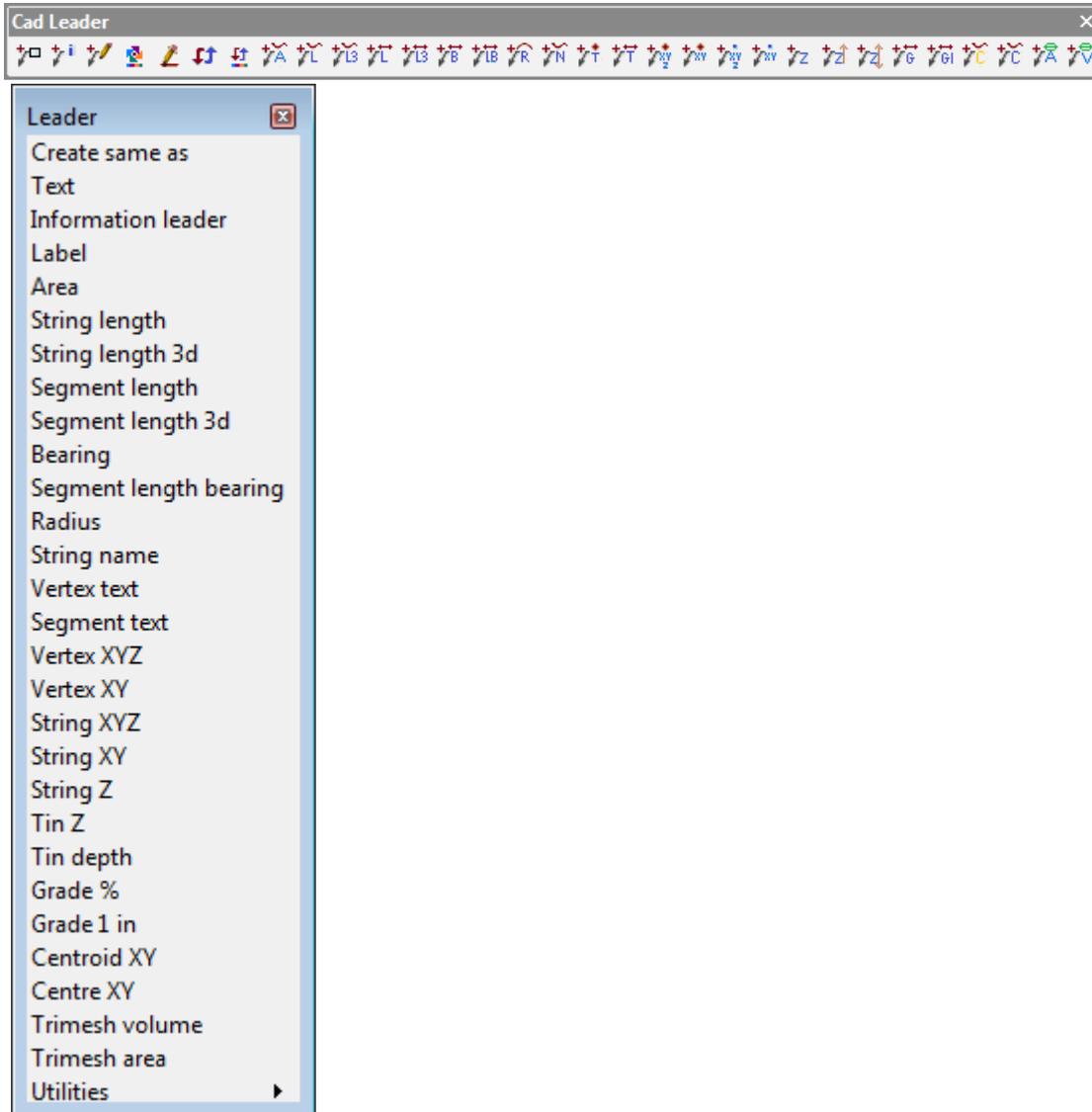
Field Description	Type	Defaults	Pop-Up
<b>Data source type</b> <i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a> in the chapter <a href="#">4 Tools and Concepts</a> .</i>		Model	
<b>Data source</b> <i>source of data to have labelled with Length dimensions.</i>	input		
<b>Association</b> <i>if <b>ticked</b>, the Length dimensions are created with Association turned ON. If <b>not ticked</b>, the Length dimensions are created with Association turned OFF.</i>	tick box	tick	
<b>Dimension point offset</b> <i>offset to use with the Length dimension. This will override the value from the <b>Style</b>.</i>	real box	5	
<b>Dimension Style</b> <i>the dimension style to use for the Length dimensions.</i>	dimension style box		
<b>Text format</b> <i>the text format to use for the Length dimensions.</i>	choice		Dim Length text format choice
<b>Output model</b> <i>model to add all the created Length dimensions to.</i>	model box		available models
<b>Create</b> <i>create Length dimensions for all the segments in the data source.</i>	button		



# 18.2 CAD Leader

Position of option on menu: CAD =>Leader

Various types of **Leaders** can be created with the **Leader** options



Apart from the Trimesh Volume and Trimesh area leaders, each type of Leader can remember the items it was created from (Association). In most cases when the items are modified, the Leader and the associated values from the items dynamically change. See [18.2.1 Leader Association](#)

All of the **Leaders** on the menu except for **Trimesh Volume** and **Trimesh Area** are also special cases of the **Information Leader**. That is, when in the **Information Leader** option, all but the Trimesh Leaders can be selected as types of the **Information Leader**.

For more information on each of the **Leader** options and menus, see

**Create same as** - create new information leader of same type as selected Information leader  
[18.2.3 Leader - Create Same As](#)

**Text** -simple leader that only displays text. See [18.2.4 Text Leader](#)

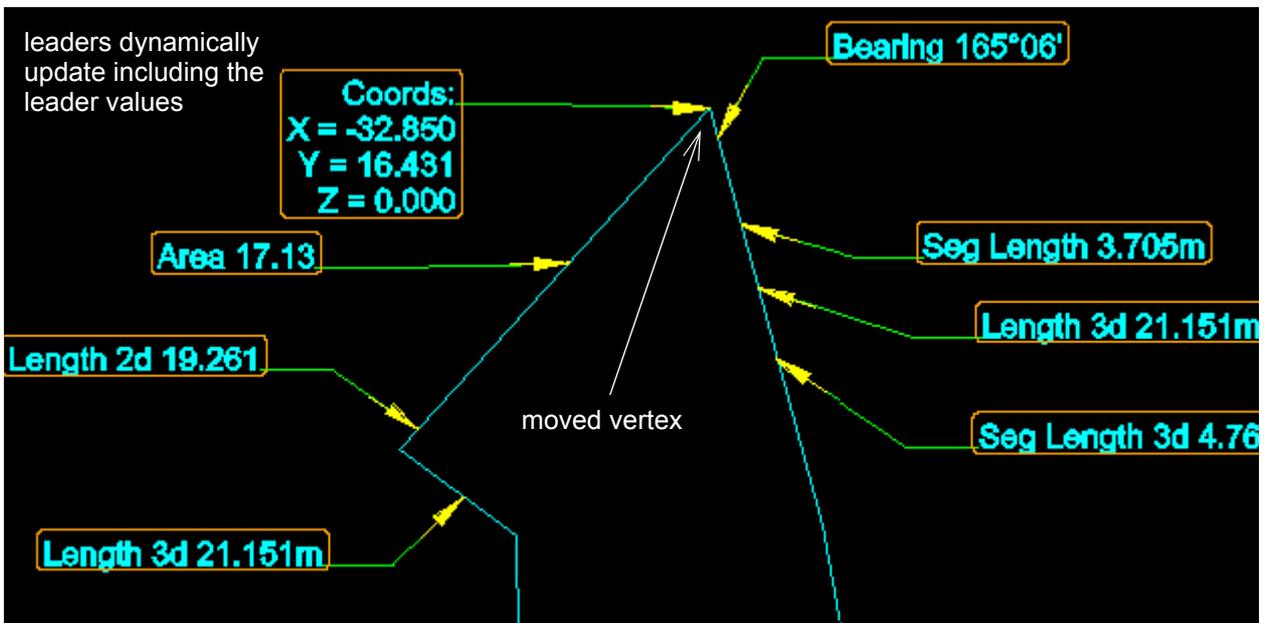
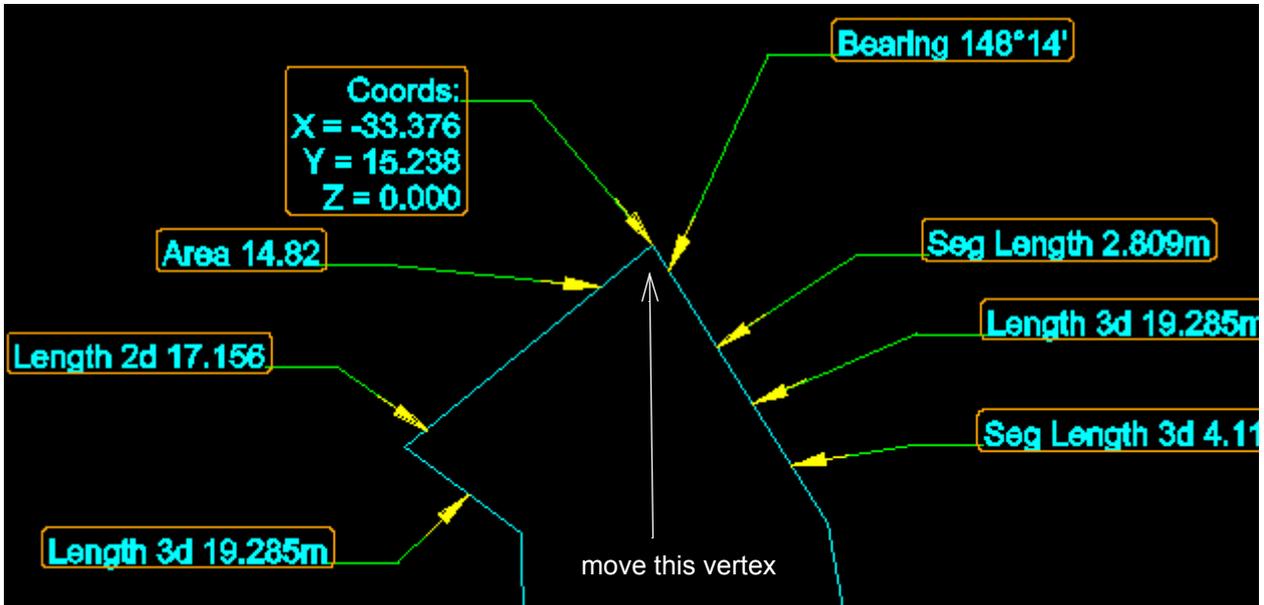
**Information Leader** - leader with many choices of values. See [18.2.5 Information Leader](#)

- Label** - a case of Information Leader that displays text. See [18.2.5.1 Label](#)
- Area** - 2D area of a selected string. See [18.2.5.2 Area](#)
- String length** - 2D length of a selected string. See [18.2.5.3 String Length](#)
- String length 3d** - 3D length of a selected string. See [18.2.5.4 String Length 3D](#)
- Segment length** - 2D length of a selected segment. See [18.2.5.5 Segment Length](#)
- Segment length 3d** - 3D length of a selected segment. See [18.2.5.6 Segment Length 3D](#)
- Bearing** - bearing of a selected string. See [18.2.5.7 Segment Bearing](#)
- Bearing & segment length** - bearing at string pick point and length of the segment. See [18.2.5.8 Bearing & Segment Length](#)
- Radius** - radius of a selected string. See [18.2.5.9 Segment Radius](#)
- String name** - name of a selected string. See [18.2.5.10 String name](#)
- Vertex text** - vertex text of a selected vertex. See [18.2.5.11 Vertex Text](#)
- Segment text** - segment text of a selected segment. See [18.2.5.12 Segment Text](#)
- Vertex XYZ** - (x,y,z) coordinates of a vertex. See [18.2.5.13 Vertex XYZ](#)
- Vertex XY** - (x,y) coordinates of a vertex. See [18.2.5.14 Vertex XY](#)
- String XYZ** - (x,y,z) coordinates of a position on a string. See [18.2.5.15 String XYZ](#)
- String XY** - (x,y) coordinates of a position on a string. See [18.2.5.16 String XY](#)
- String Z** - z coordinates (level) of a position on a string. See [18.2.5.18 Tin Z](#)
- Tin Z** - z value of tin at a selected (x,y) position. See [18.2.5.18 Tin Z](#)
- Tin depth** - depth from a string position to a tin. See [18.2.5.19 Tin Depth](#)
- Grade %** - grade as a percentage at a point on a string. See [18.2.5.20 Grade \(%\)](#)
- Grade 1 in** - grade as 1 in (i.e. slope) at a point on a string. See [18.2.5.21 Grade 1 n \(Slope\)](#)
- Centroid XY** - (x,y) coordinates of selected string. See [18.2.5.22 Centroid XY](#)
- Centre XY** - (x,y) coordinates of medial centre of selected string. See [18.2.5.23 Medial Centre XY](#)
- Trimesh volume** - volume of a closed trimesh. See [18.2.5.24 Trimesh Volume](#)
- Trimesh area** - surface area of one side of a trimesh. See [18.2.5.25 Trimesh Surface Area](#)
- Utilities** - various leader utilities. See [18.2.6 Leader Utilities](#)

## 18.2.1 Leader Association

When *leaders* are created, the leader can remember the items that were used to create them (**Association**).

In most cases when items are moved, the leaders associated with the items dynamically change.

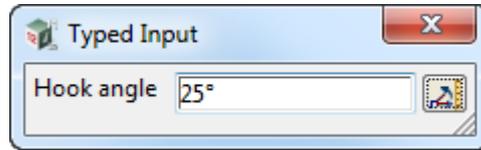


In cases where it is not automatic, there is a **Recalc** and **Recalc all** for leaders and dimensions.

In the leader options, typing a or A toggles Association **ON** and **OFF**.

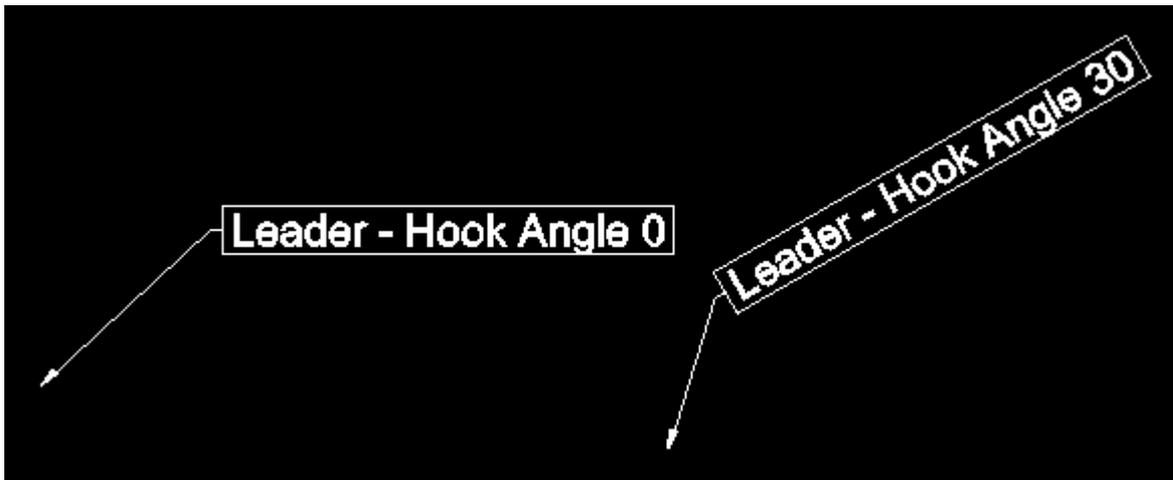
## 18.2.2 Hook Angle

Typing **h** or **H** allows you to change the **Hook Angle** and brings up the *Hook angle Typed Input* box.



The new hook angle is typed into the box and **<Enter>** pressed.

The units for hook angle is degrees entered in hp notation, and is measure in a counter clockwise direction from the positive x-axis.



## 18.2.3 Leader - Create Same As

**Create same as** creates a new Leader of the same type as a selected Leader.

### Step 1.

Select **Create same as** and the following message is written to the screen message area

```
<[Pick leader]> [picks][fast][Menu]
```

### Step 2. Select an existing Leader

When an existing **Leader** is selected, a new **create Leader** of the same Leader type as the selected Leader, is started.

## 18.2.4 Text Leader

This is simplest type of Leader and only creates a leader with user defined text.

**Text Leader** creates a leader line from a selected point (the arrow point) to a selected hook point, with user given text drawn at the end of the leader line.

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

### Step 1. Pick the arrow point (the point to start the Leader from)

Select Leader and the following message is written to the screen message area

```
<[Pick arrow point] (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][
```

If Association is **ON** then the start of the Leader is associated with the selected arrow point.

### Step 2. Pick the hook point

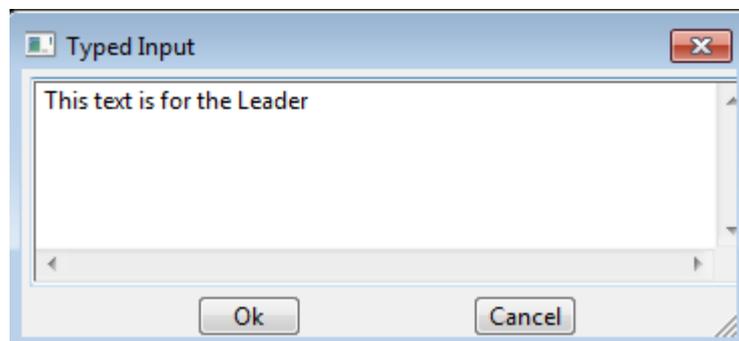
The following message is written to the screen message area.

```
<Pick hook point> [picks][fast][Menu]
```

If Association is **ON** then the hook point of the leader is associated with the selected hook point.

### Step 3.

The text **Typed Input** box then comes up for the text for the leader to be typed into.



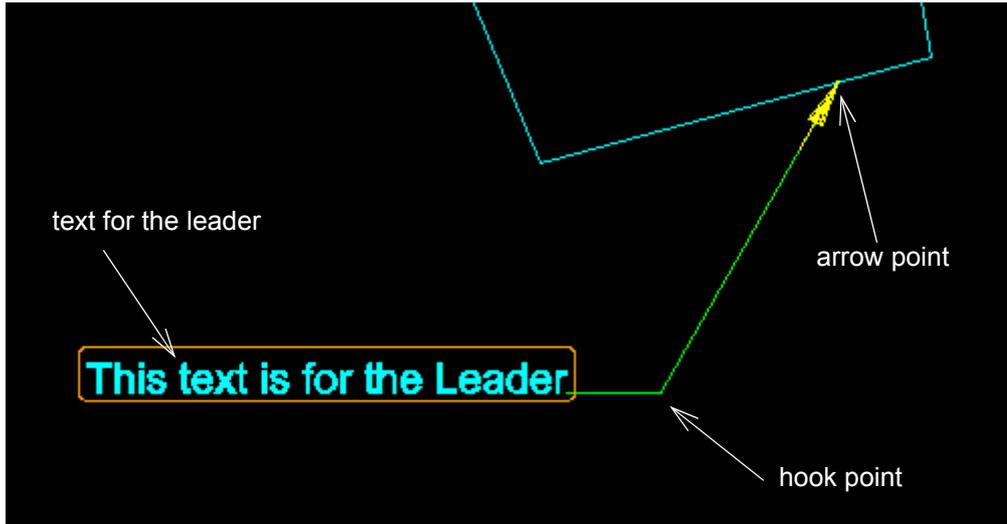
Whilst you are in the **Typed Input** box the following message is written to the screen message area.

```
Enter Text [Caret][Same as][Menu] select a button
```

The text for the Leader is typed into the **Typed Input** box and **OK** clicked.

The *Leader* is then created with the current hook angle.

The **Leader** option then starts again. That is, goes back to **Step 1**.



For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

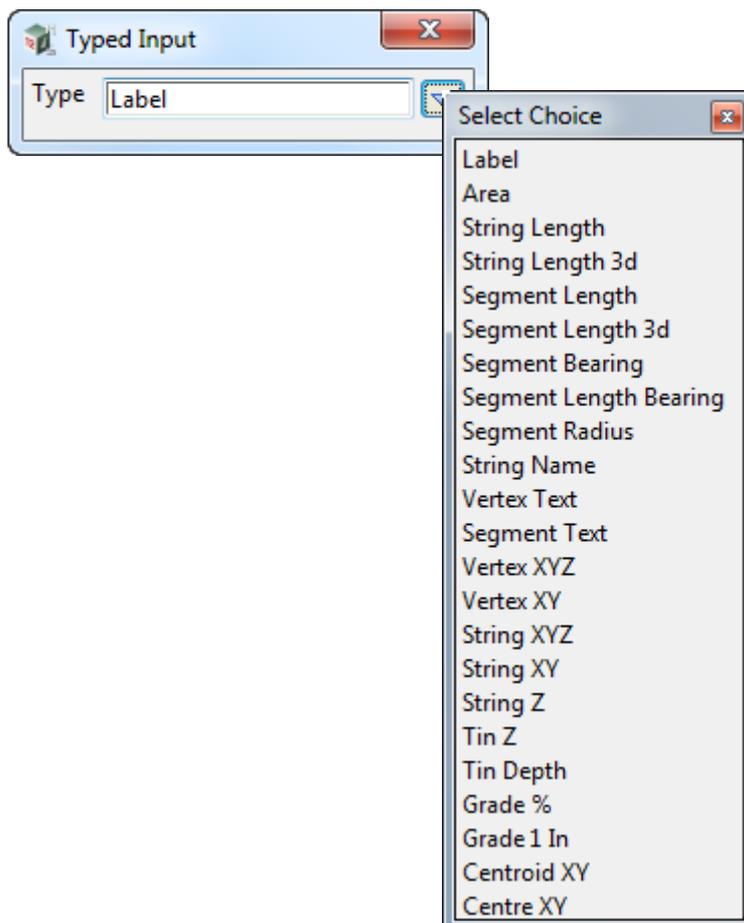
## 18.2.5 Information Leader

**Information Leader** creates a leader line from a selected point (the **arrow point**) to a selected **hook point**, with a user selected choice of what information is shown at the end of the leader line.

When **Information Leader** is selected, the following message is written to the screen message area.

```
<(a)ssociative (t)ype[Label] Pick arrow location (s)tyle[Arial Border] (h)ook-angle[25°] > [picks][fast][Menu]
```

Typing **t** or **T** brings up the **Information Leader Type Typed Input** box which has a choice pop up for selecting what information is drawn at the end of the leader.



After a selection is made, the message changes to indicate the current type of **Information Leader** being created. For example, for the choice **Area**

type of Information Leader being created

```
< (t)ype[Area] Pick string (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] fixed-a(r)row > [picks][fast][Menu]
```

For more information on each of the types for Information Leader see

**Label** - simple leader that only displays text. See [18.2.5.1 Label](#)

**Area** - 2D area of a selected string. See [18.2.5.2 Area](#)

**String length** - 2D length of a selected string. See [18.2.5.3 String Length](#)

- String length 3d** - 3D length of a selected string. See [18.2.5.4 String Length 3D](#)
- Segment length** - 2D length of a selected segment. See [18.2.5.5 Segment Length](#)
- Segment length 3d** - 3D length of a selected segment. See [18.2.5.6 Segment Length 3D](#)
- Bearing** - bearing of a selected string. See [18.2.5.7 Segment Bearing](#)
- Bearing & segment length** - bearing at string pick point and length of the segment. See [18.2.5.8 Bearing & Segment Length](#)
- Radius** - radius of a selected string. See [18.2.5.9 Segment Radius](#)
- String name** - name of a selected string. See [18.2.5.10 String name](#)
- Vertex text** - vertex text of a selected vertex. See [18.2.5.11 Vertex Text](#)
- Segment text** - segment text of a selected segment. See [18.2.5.12 Segment Text](#)
- Vertex XYZ** - (x,y,z) coordinates of a vertex. See [18.2.5.13 Vertex XYZ](#)
- Vertex XY** - (x,y) coordinates of a vertex. See [18.2.5.14 Vertex XY](#)
- String XYZ** - (x,y,z) coordinates of a position on a string. See [18.2.5.15 String XYZ](#)
- String XY** - (x,y) coordinates of a position on a string See [18.2.5.16 String XY](#)
- String Z** - z coordinates (level) of a position on a string See [18.2.5.17 String Z](#)
- Tin Z** - z value of tin at a selected (x,y) position. See [18.2.5.18 Tin Z](#)
- Tin depth** - depth from a string position to a tin. See [18.2.5.19 Tin Depth](#)
- Grade %** - grade in percent at a point on a string. See [18.2.5.20 Grade \(%\)](#)
- Grade 1 in** - grade as 1 in (i.e. slope) at a point on a string. See [18.2.5.21 Grade 1 n \(Slope\)](#)
- Centroid XY** - (x,y) coordinates of selected string. See [18.2.5.22 Centroid XY](#)
- Centre XY** - (x,y) coordinates of medial centre of selected string. See [18.2.5.23 Medial Centre XY](#)

### 18.2.5.1 Label

**Information Leader** choice **Label** displays in the leader box, user given text.

**Label** creates a leader line from a selected point (the pick point) to a selected hook point, with the given text drawn at the end of the leader line.

The leader is added to the model given in the CAD toolbar.

**Note:** Unlike the **Text Leader** which is also used for placing text and brings up a Typed Input box for entering multiline text (see [18.2.4 Text Leader](#)), **Label** is a single line format which needs new line characters if a new line is needed.

The case for using **Label** instead of the **Text** leader is when you have the same text to place multiple times. The **customise (t)ext** command uses the last specified text, and also remembers the last twelve of them.

#### Step 1. Pick the point to start the Leader from

Select **Leader** and the following message is written to the screen message area

```
< (t)ype[Label] Pick arrow location (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the selected pick point.

#### Step 2. Pick the hook point

The following message is written to the screen message area.

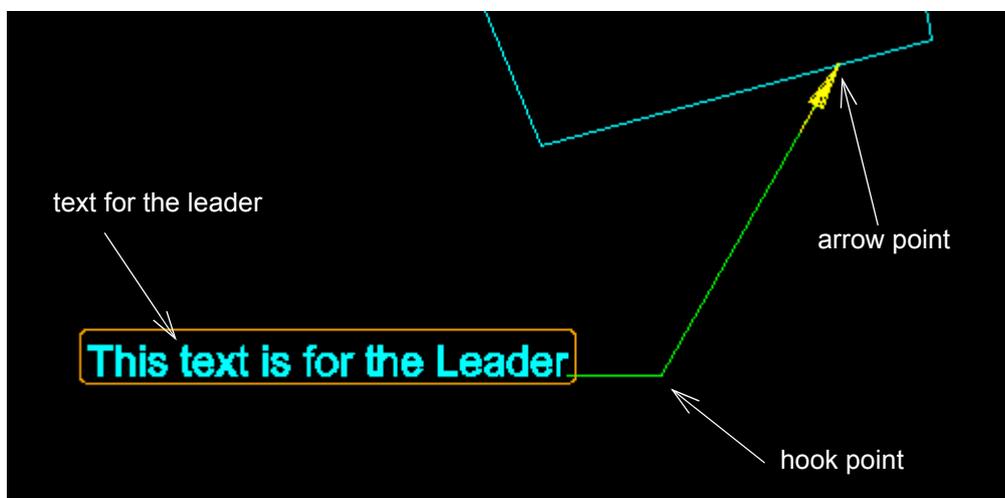
```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

Unlike the **Text Leader** which brings up a Typed Input box for entering multiline text (see [18.2.4 Text Leader](#)), this is a single line format which needs new line characters if a new line is needed.

When the hook point is selected, the *Leader* is created.

The **Label Information Leader** then starts again. That is, it goes back to **Step 1**.



For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.2 Area

**Information Leader** choice **Area** displays in the leader box, the 2D (plan) area of a selected string. If the selected string is not closed, the area is calculated for the polygon formed by joining the first and the last vertices.

A leader line is drawn from either selected position on a string (string pick point) or the string centre, to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the area is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with its 2D (plan) area

When **Area** is the choice, the following message is written to the screen message area

```
< (t)ype[Area] Pick string (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] fixed-a(r)row> [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

**r** or **R** toggles between the arrow starting at the pick point (**free arrow**) or at the polygon centre (**fixed arrow**)

If Association is **ON** then the start of the Leader is associated with the string pick point or the polygon centre (depending on **r**).

#### Step 2. Pick the hook point

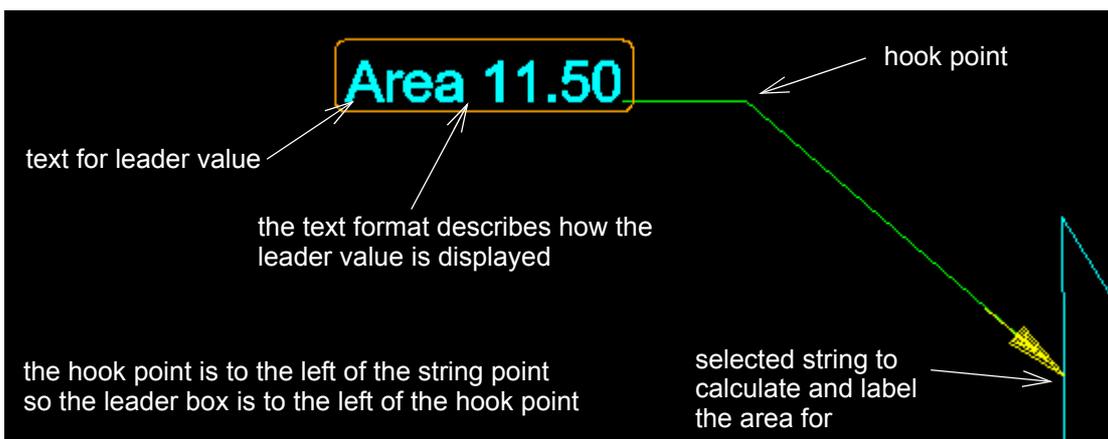
The following message is written to the screen message area.

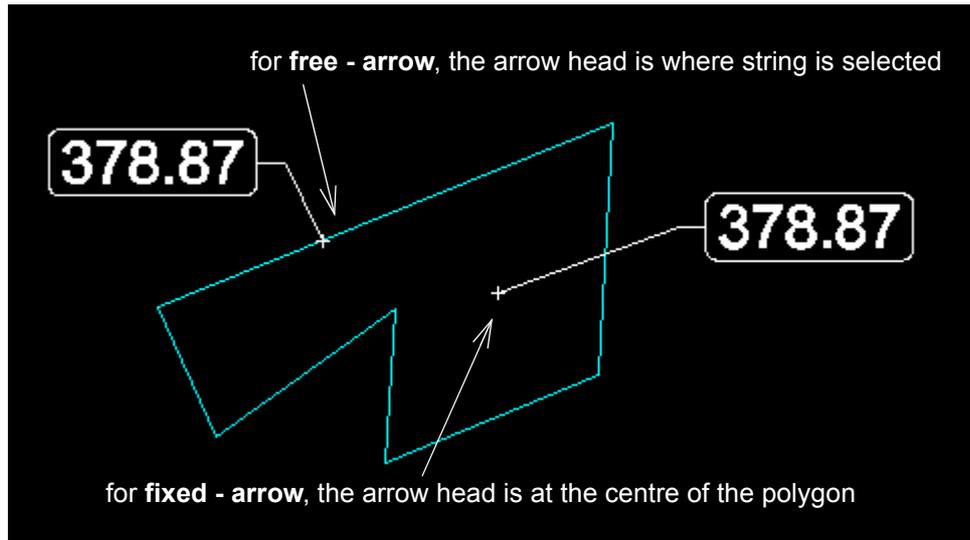
```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Area Information Leader** then starts again. That is, it goes back to **Step 1**.





#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.3 String Length

**Information Leader** choice **String length** displays in the leader box, the 2D (plan) length of a selected string.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the string length is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with its 2D length

When **String Length** is the choice, the following message is written to the screen message area

```
< (t)ype[String Length] Pick string (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

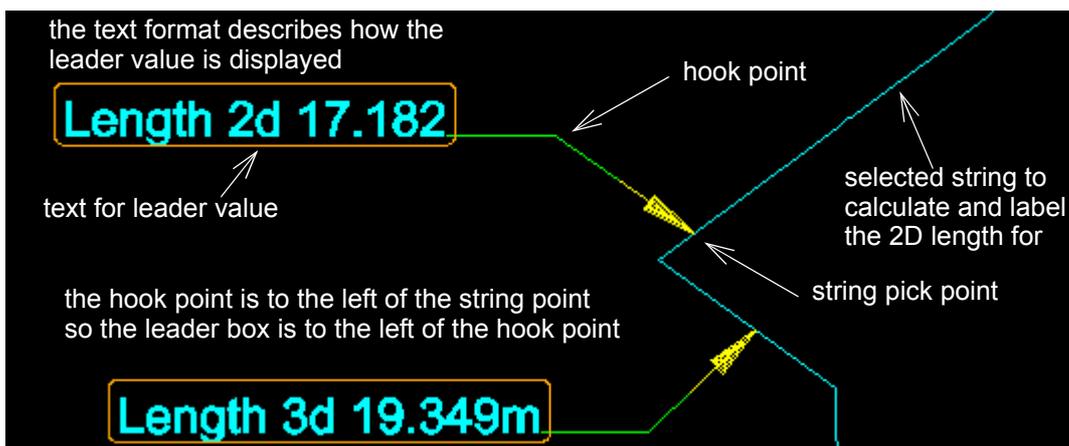
The following message is written to the screen message area.

```
< Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **String Length Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.4 String Length 3D

**Information Leader** choice **String length 3d** displays in the leader box, the 3D length of a selected string.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the length is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with its 3D length

When **String length 3d** is the choice, the following message is written to the screen message area

```
< (t)ype[String Length 3d] Pick string (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

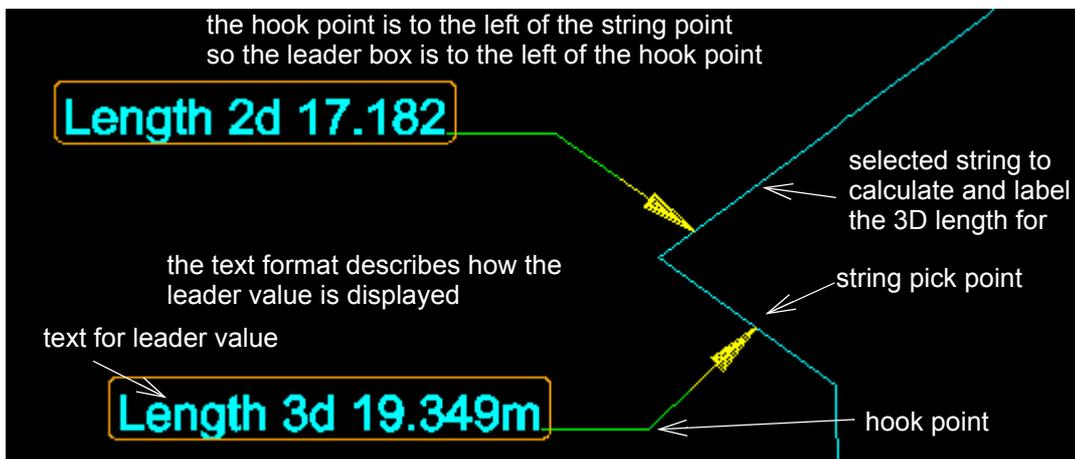
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **String Length 3D Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.5 Segment Length

**Information Leader** choice **Segment length** displays in the leader box, the 2D (plan) length of a selected **segment**.

A leader line is drawn from a selected position on a segment (segment pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the segment length is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the segment to label with its 2D length

When **Segment Length** is the choice, the following message is written to the screen message area

```
< (t)type[Segment Length] Pick segment (a)ssociative (s)style[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the segment pick point.

#### Step 2. Pick the hook point

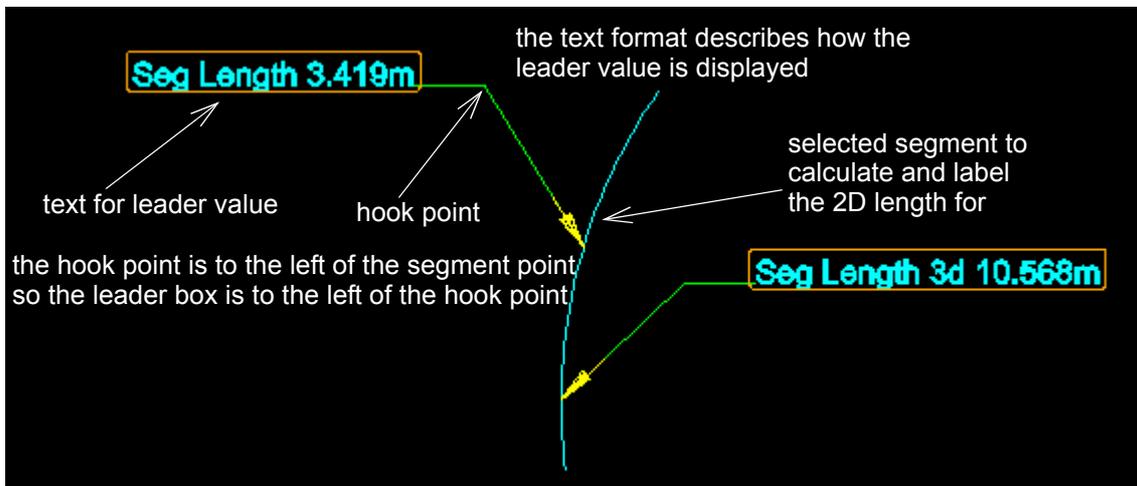
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Segment Length Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.6 Segment Length 3D

**Information Leader** choice **Segment length 3d** displays in the leader box, the 3D length of a selected segment.

A leader line is drawn from the pick position on a selected segment (segment pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the 3d length of the segment is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the segment to label with its 3D length

When **Segment Length 3d** is the choice, the following message is written to the screen message area

```
< (t)ype[Segment Length 3d] Pick segment (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the segment pick point

#### Step 2. Pick the hook point

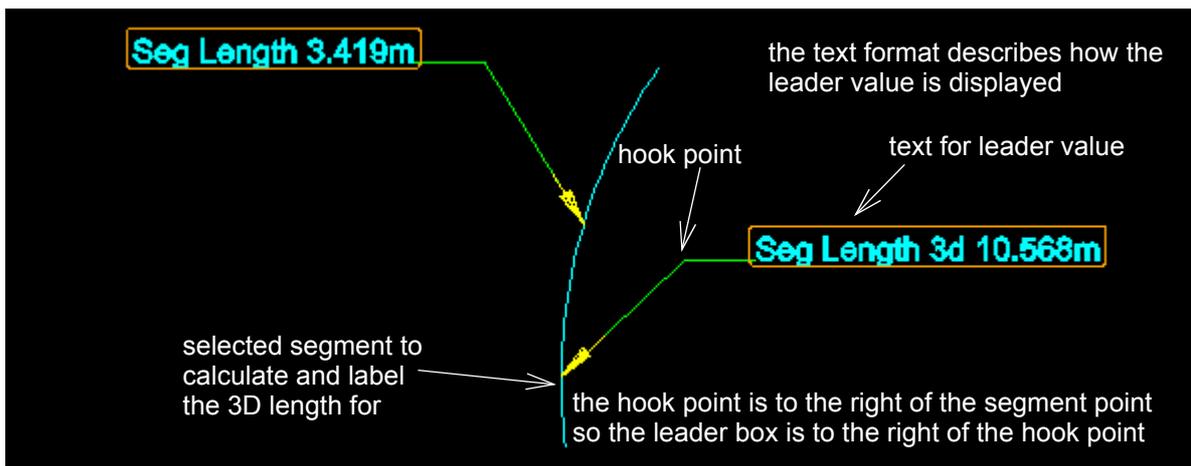
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Segment Length 3D Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.7 Segment Bearing

**Information Leader** choice **Segment Bearing** displays in the leader box, the bearing of a selected position on string (string pick point).

A leader line is drawn from the string pick point to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the bearing at the string pick point is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the segment to label with its bearing

When **Segment Bearing** is the choice, the following message is written to the screen message area

```
< (t)ype[Segment Bearing] Pick point on segment (a)ssociative (s)tyl[eArial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

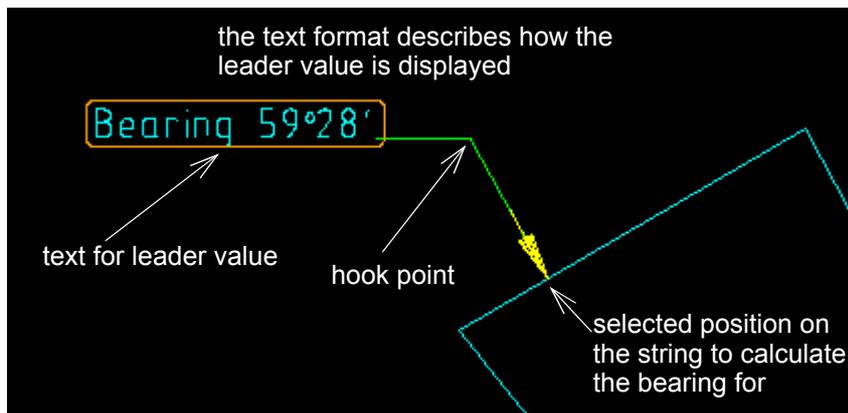
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Segment Bearing Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.8 Bearing & Segment Length

**Information Leader** choice **Bearing & segment length** displays in the leader box, the bearing at the string pick point and the length of the segment it is on.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the bearing and length of the pick point and segment is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with its bearing & length

When **Segment Length & Bearing** is the choice, the following message is written to the screen message area

```
< (t)ype[Segment Length Bearing] Pick point on segment (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the segment pick point.

#### Step 2. Pick the hook point.

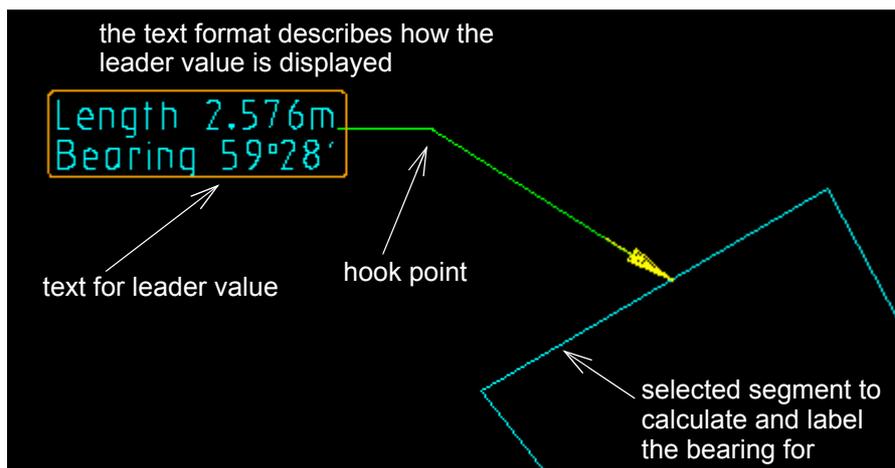
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Segment Length & Bearing Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.9 Segment Radius

**Information Leader** choice **Segment Radius** displays in the leader box the radius of a selected segment. If the segment is a straight then the radius is displayed as 0. If the segment is a transition, then the displayed radius is the radius of the transition at the picked position.

A leader line is drawn from a string pick point to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the radius of the string pick point is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the segment to label with its radius

When **Segment Radius** is the choice, the following message is written to the screen message area

```
< (t)ype[Segment Radius] Pick point on segment (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

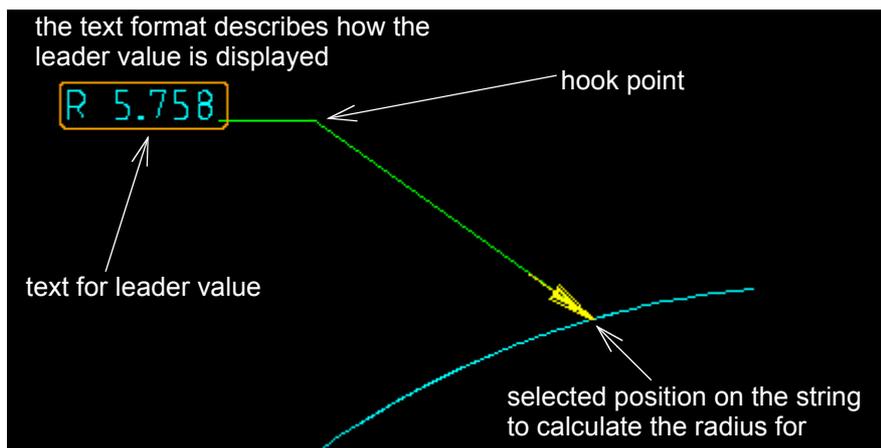
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

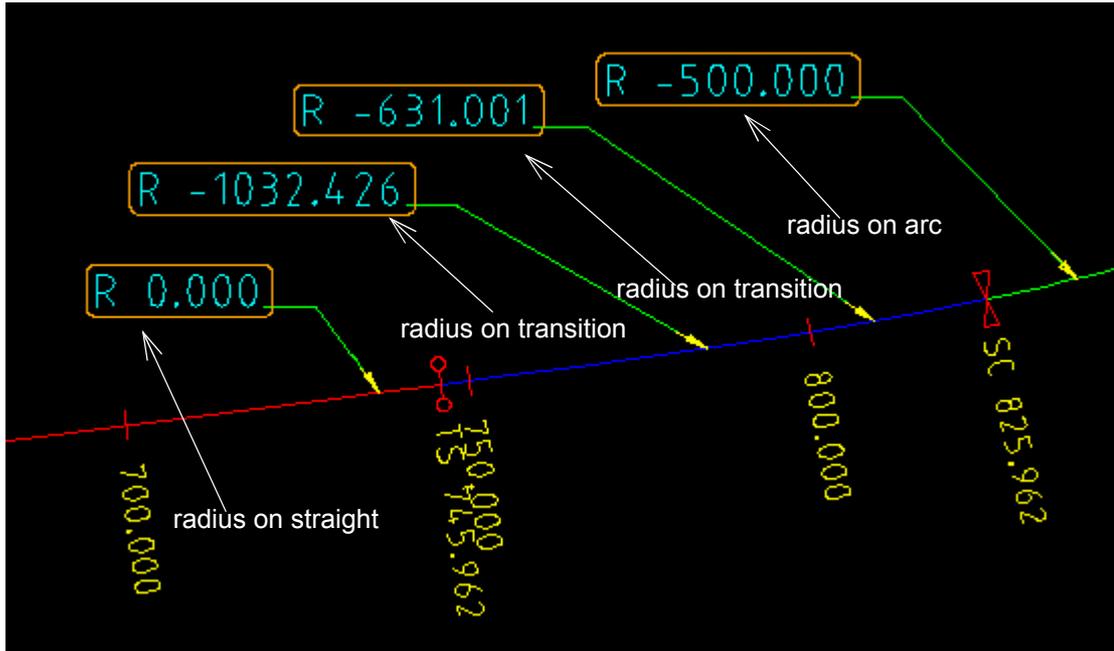
The **Segment Radius Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.



For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.10 String name

**Information Leader** choice **String name** displays in the leader box, the name of a selected string.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the name of the string is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with its name

When **String Name** is the choice, the following message is written to the screen message area

```
< (t)type[String Name] Pick string (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the segment pick point.

#### Step 2. Pick the hook point

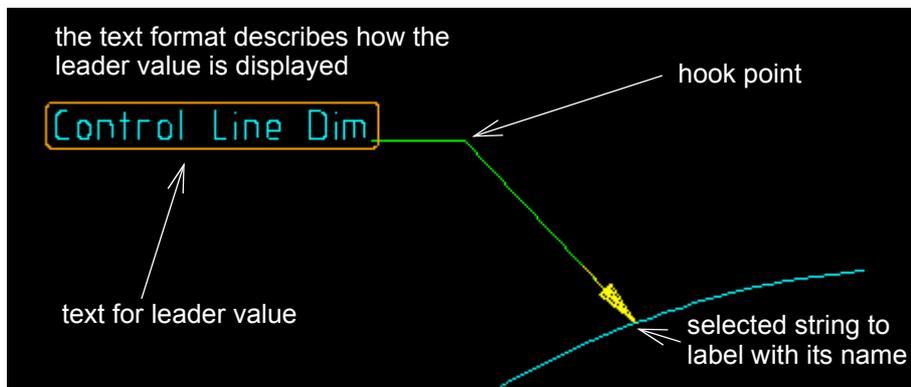
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **String Name Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.11 Vertex Text

**Information Leader** choice **Vertex text** displays in the leader box, the vertex text of a selected string vertex.

A leader line is drawn from a selected vertex on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the vertex text for the selected vertex of the string is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string vertex to label with its vertex text

When **Vertex Text** is the choice, the following message is written to the screen message area

```
< (t)ype[Vertex Text] Pick vertex (a)ssociative (s)tyl[eArial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the segment pick point.

#### Step 2. Pick the hook point

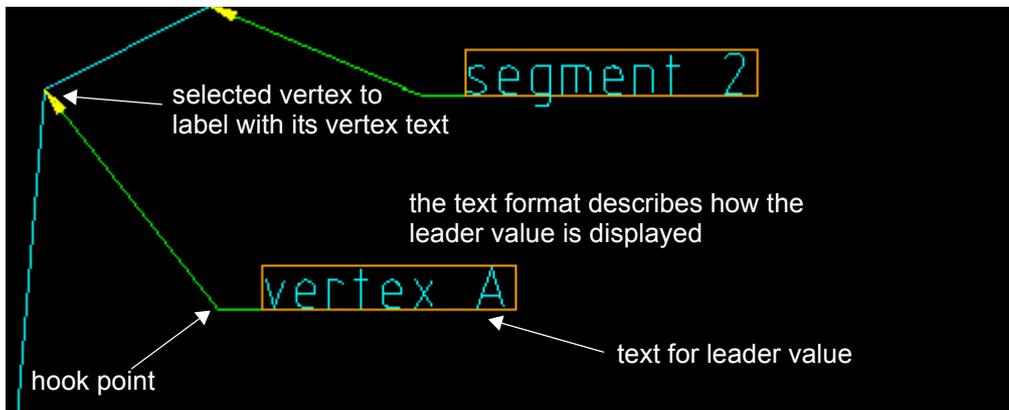
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Vertex Text Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.12 Segment Text

**Information Leader** choice **Segment text** displays in the leader box, the segment text of a selected string segment.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the segment text for the selected segment of the string is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string segment to label with its segment text

When **Segment Text** is the choice, the following message is written to the screen message area

```
< (t)ype[Segment Text] Pick segment (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the segment pick point.

#### Step 2. Pick the hook point

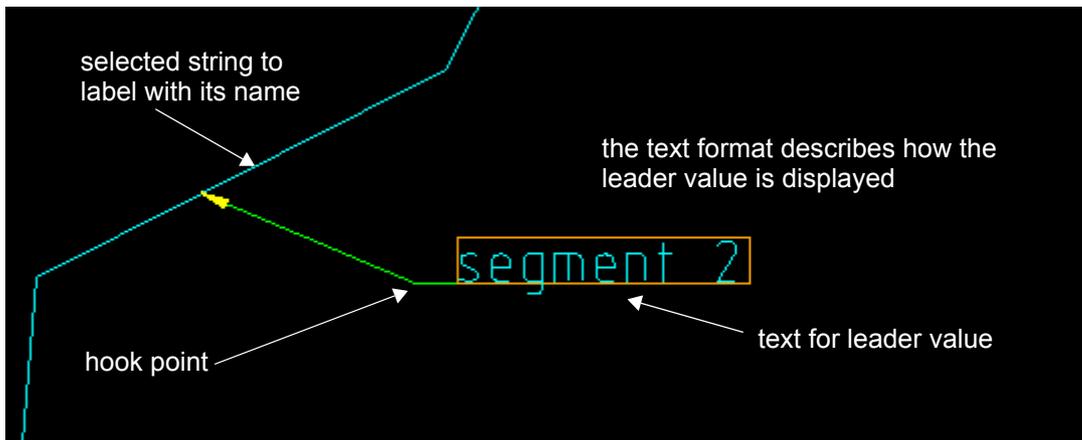
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Segment Text Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.13 Vertex XYZ

**Information Leader** choice **Vertex XYZ** displays in the leader box, the x, y and z coordinates of a selected vertex.

A leader line is drawn from a selected vertex on a string to a selected hook point, and then to the end of the hook itself.

Using the text format, the x, y and z coordinates of a selected vertex are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the vertex to label with its x, y and z coordinates

When **Vertex XYZ** is the choice, the following message is written to the screen message area

```
< (t)ype[Vertex XYZ] Pick vertex (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the vertex.

#### Step 2. Pick the hook point

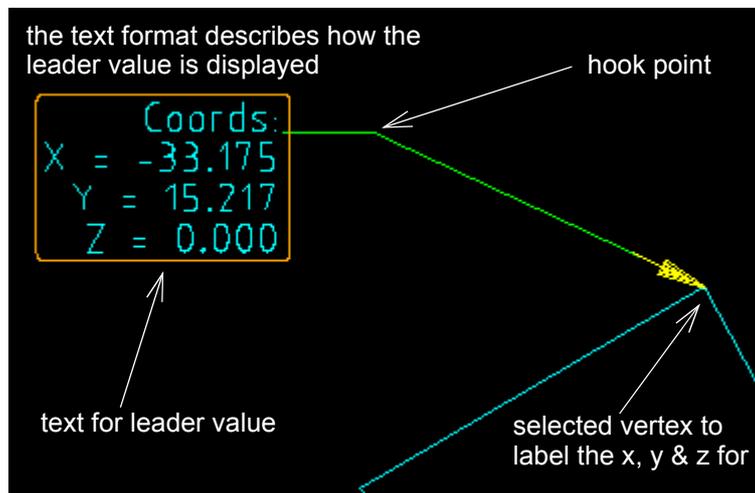
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Vertex XYZ Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.14 Vertex XY

**Information Leader** choice **Vertex XY** displays in the leader box, the x and y coordinates of a selected vertex.

A leader line is drawn from a selected vertex on a string to a selected hook point, and then to the end of the hook itself.

Using the text format, the x and y coordinates of a selected vertex are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the vertex to label with its x and y coordinates

When **Vertex XY** is the choice, the following message is written to the screen message area

```
< (t)ype[Vertex XY] Pick vertex (a)ssociative (s)tyl[eArial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the vertex.

#### Step 2. Pick the hook point

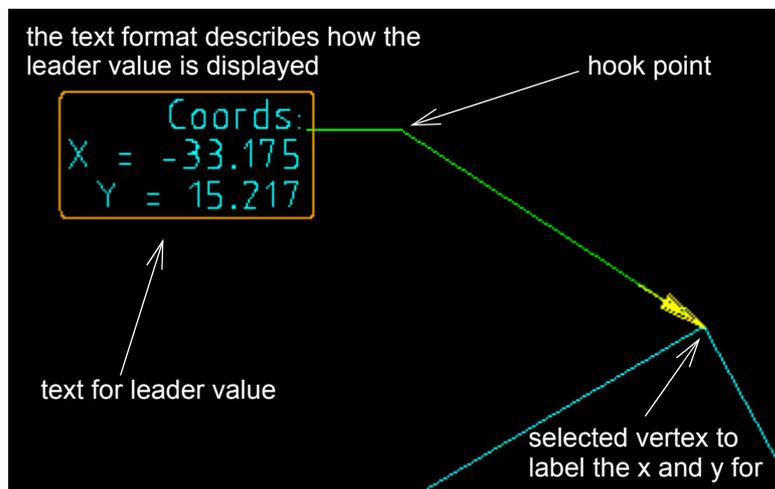
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Vertex XY Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.15 String XYZ

**Information Leader** choice **String XYZ** displays in the leader box, the x, y and z coordinates of any selected position on a string.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the x, y and z coordinates of the selected string position are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1.Pick the string position to label with its x, y and z coordinates

When **String XYZ** is the choice, the following message is written to the screen message area

```
< (t)ype[String XYZ] Pick point (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2.Pick the hook point

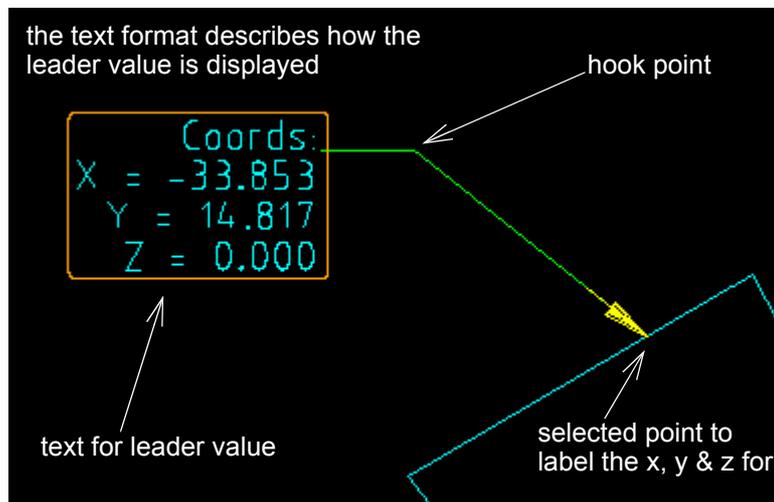
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **String XYZ Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.16 String XY

**Information Leader** choice **String XY** displays in the leader box, the x and y coordinates of any selected position on a string.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the x and y coordinates of the selected string position are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string position to label with its x and y coordinates

When **String XY** is the choice, the following message is written to the screen message area

```
< (t)ype[String XY] Pick point (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

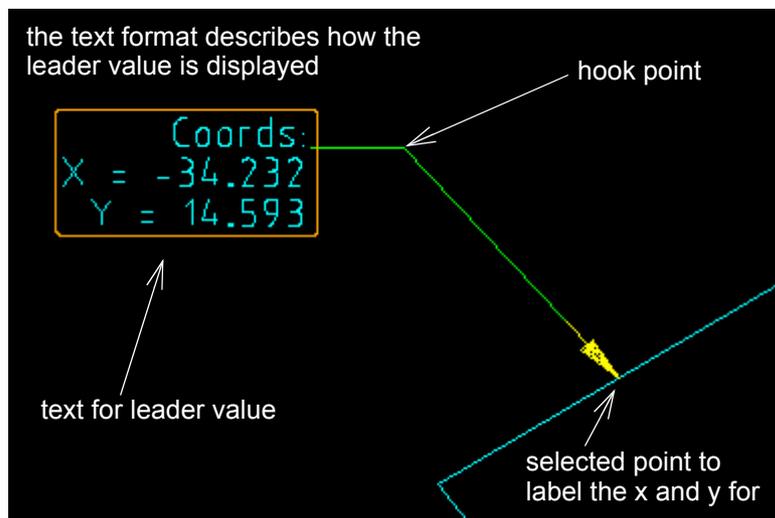
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **String XY Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.17 String Z

**Information Leader** choice **String Z** displays in the leader box, the z coordinate (level) of any selected position on a string.

A leader line is drawn from a selected position on a string (string pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the z coordinate (level) of the selected string position are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string position to label with level (z coordinate)

When **String Z** is the choice, the following message is written to the screen message area

```
< (t)ype[String Z] Pick point (a)ssociative (s)tyl[e[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

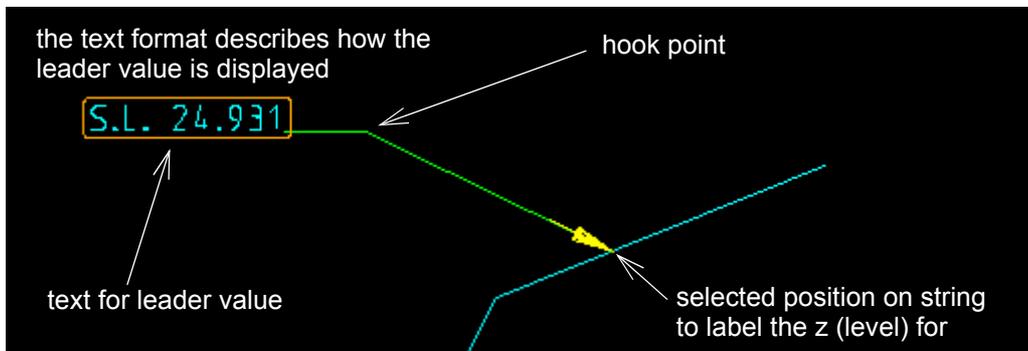
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **String Z Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the segment pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the segment pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.18 Tin Z

**Information Leader** choice **Tin Z** displays in the leader box, the z value (level) from the tin at the (x,y) coordinates of a selected position.

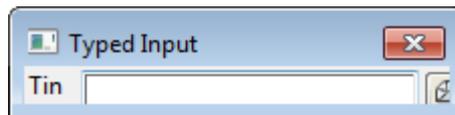
A leader line is drawn from the selected position (pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the level from the tin is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the position that is to be labelled with the level (z value) from the tin

When **Tin Z** is the choice, and if it is the first time the **Tin z** option has been used in this session, then the *Tin Typed Input* box is placed on the screen.



The tin to take the levels from is picked from the pop up list, or typed into the box and **<Enter>** pressed. The box is then removed.

The following message is written to the screen message area.

```
< (t)ype[Tin Z] Pick point on tin (a)ssociative (s)tyl[eArial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

**i** or **I** brings up the *Tin Typed Input* box to select a new tin.

If Association is **ON** then the start of the Leader is associated with the pick point

#### Step 2.

The following message is written to the screen message area.

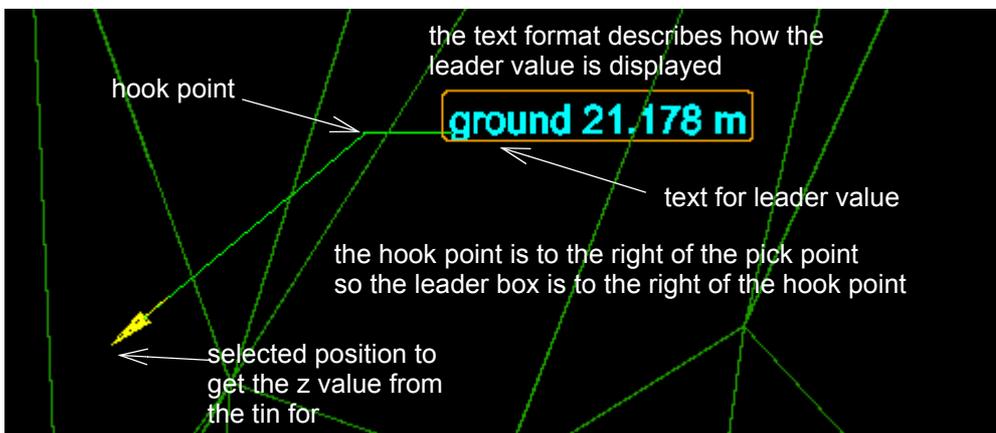
```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

#### Pick the hook point.

The *Information Leader* is then created.

The **Tin Z Information Leader** then starts again. That is, it goes back to **Step 1**.



**Note** - the tin itself does not have to be on any view.

#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.19 Tin Depth

**Information Leader** choice **Tin Depth** displays in the leader box, the depth at a selected position on a string to a tin.

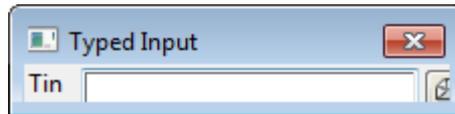
A leader line is drawn from the selected string position (pick point) to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the depth from the pick point to the tin is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the point that is to be labelled with its depth to the tin

When **Tin Depth** is the choice, and if it is the first time the **Tin z** option has been used in this session, then the **Tin Typed Input** box is placed on the screen.



The tin to take the levels from is picked from the pop up list, or typed into the box and **<Enter>** pressed. The box is then removed

The following message is written to the screen message area.

```
< (t)ype[Tin Depth] Pick point on tin (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

**i** or **I** brings up the **Tin Typed Input** box to select a new tin.

If Association is **ON** then the start of the Leader is associated with the pick point

#### Step 2. Pick the hook point

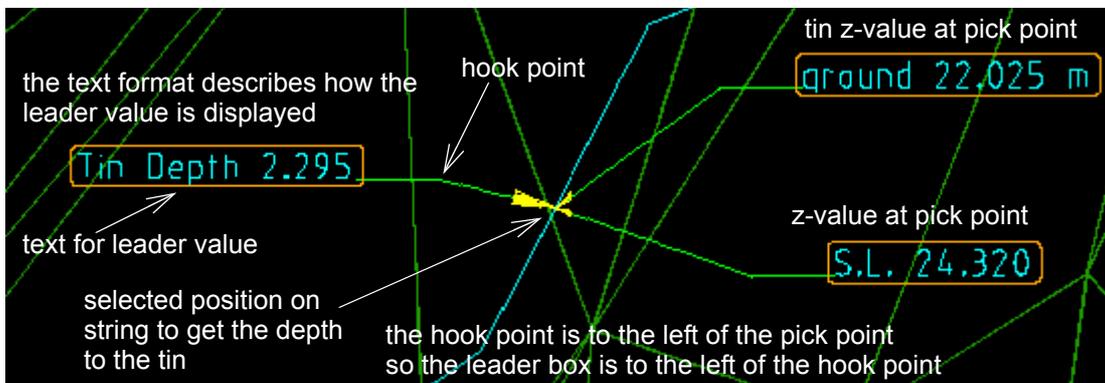
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The **Information Leader** is then created.

The **Tin Depth Information Leader** then starts again. That is, it goes back to **Step 1**.



**Note** - the tin itself does not have to be on any view.

#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.20 Grade (%)

**Information Leader** choice **Grade (%)** displays in the leader box, the percent grade of a selected position on string (string pick point).

A leader line is drawn from the string pick point to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the percent grade at the string pick point is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the position on a string to label with per cent grade

When **Grade per cent** is the choice, the following message is written to the screen message area

```
< (t)ype[Grade %] Pick point on string (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2. Pick the hook point

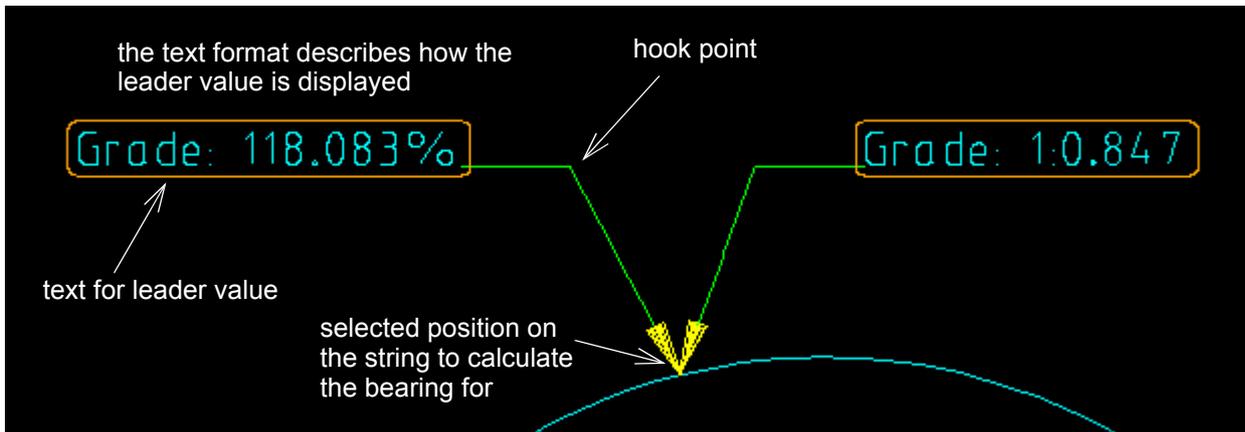
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Grade per cent Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.21 Grade 1 n (Slope)

**Information Leader** choice **Grade 1 in** displays in the leader box, the grade as a 1 in (slope) of a selected position on string (string pick point).

A leader line is drawn from the string pick point to a selected hook point, and then to the end of the hook itself.

Using the text format, the value of the grade as a 1 in (slope) at the string pick point is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1.Pick the position on a string to label with its grade as 1:in (slope)

When **Grade 1 in** is the choice, the following message is written to the screen message area

```
< (t)type[Grade 1 In] Pick point on string (a)ssociative (s)tyl[Arial Border] (h)ook-angle[0°] > [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#).  
**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#).  
**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#).  
**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#).

If Association is **ON** then the start of the Leader is associated with the string pick point.

#### Step 2.Pick the hook point

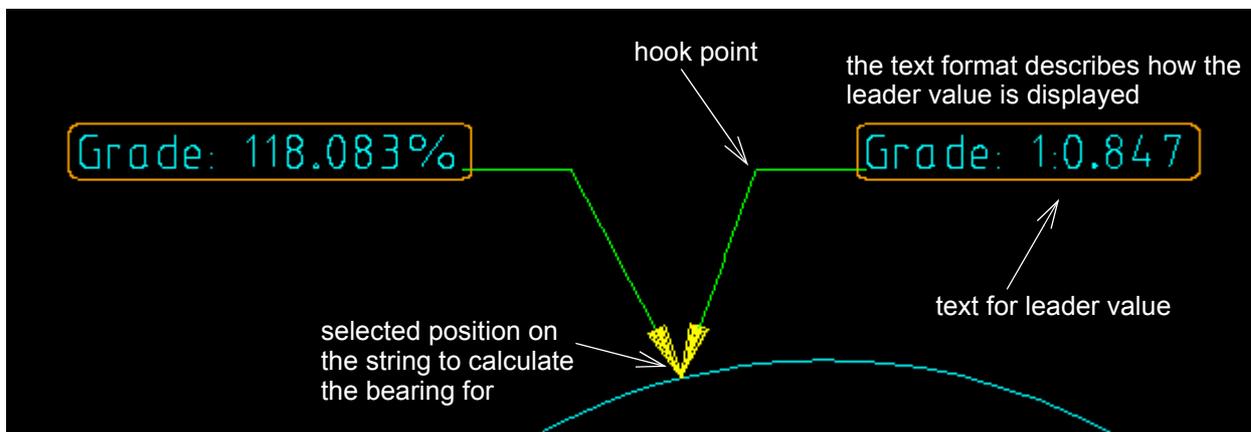
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Grade 1 in Information Leader** then starts again. That is, it goes back to **Step 1**.



#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the string pick point then the leader box is drawn to the left of the hook point. If the hook point is to the right of the string pick point then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.22 Centroid XY

**Information Leader** choice **Centroid XY** displays in the leader box, the (x,y) coordinates of the centroid of a selected string. If the string is not a closed string (a polygon) then it is temporarily closed for calculating the Centroid. Note that the centroid may be outside the closed string - for a point that is inside the string, see [18.2.5.23 Medial Centre XY](#).

A string is selected (string pick point) and a position for the hook and leader line is drawn from either the string pick point or the **centroid** of the string, to a selected hook point, and then to the end of the hook itself.

Using the text format, the (x,y) coordinates of the centroid of the selected string are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with the (x,y) coordinates of its centroid

When **Centroid XY** is the choice, the following message is written to the screen message area

```
< (t)ype[Centroid XY] Pick string (a)ssociative (s)tyle[Arial Border] (h)ook-angle[0°] fixed-a(r)row> [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

**r** or **R** toggles between the arrow starting at the pick point (**free arrow**) or at the polygon centre (**fixed arrow**)

If Association is **ON** then the start of the Leader is associated with the pick point or the centroid of the string.

#### Step 2. Pick the hook point

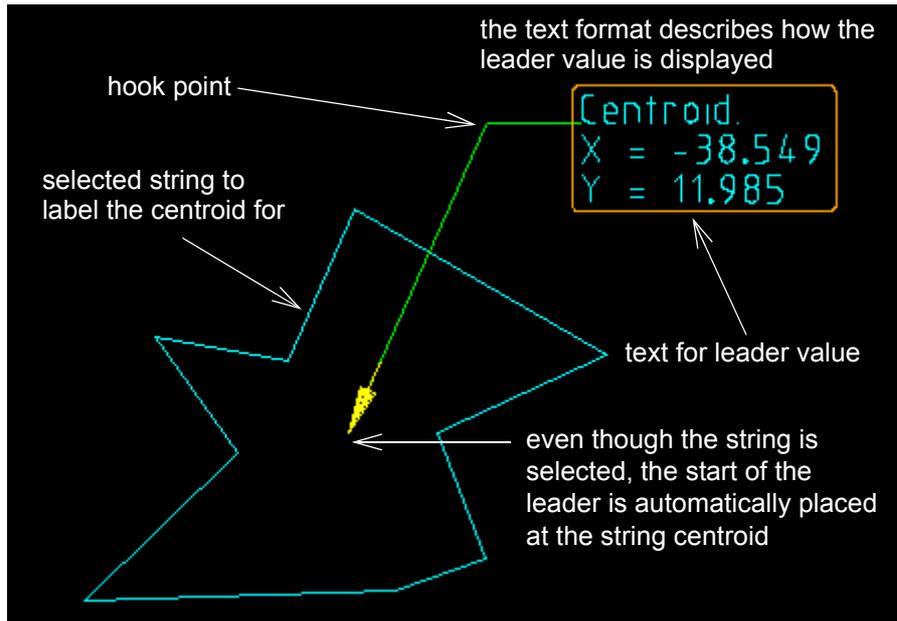
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Centroid XY Information Leader** then starts again. That is, it goes back to **Step 1**.



**Note on Position of Leader Box With Respect to the Hook Point**

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the centroid of the string then the leader box is drawn to the left of the hook point. If the hook point is to the right of the centroid of the string then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.23 Medial Centre XY

**Information Leader** choice **Centre XY** displays in the leader box, the (x,y) coordinates of the medial axis centre of a selected string. If the string is not a closed string (polygon) then it is temporarily closed for calculating the Medial centre.

**Note** - the Medial centre is a point calculated by 12d to use as the “centre” of a polygon and unlike a centroid, it is always **inside** the polygon. **WARNING** - this currently does not take arcs into consideration.

A string is selected (string pick point) and a position for the hook and leader line is drawn from either the string pick point or the **medial axis centre** of the string, to a selected hook point, and then to the end of the hook itself.

Using the text format, the (x,y) coordinates of the medial centre of the selected string are displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the string to label with the (x,y) coordinates of its (medial) centre

When **Centre XY** is the choice, the following message is written to the screen message area

```
< (t)ype[Centroid XY] Pick string (a)ssociative (s)tyl[eArial Border] (h)ook-angle[0°] fixed-a(r)row> [picks][fast][Menu]
```

**t** or **T** changes the type of Information Leader. See [18.2.5 Information Leader](#)

**a** or **A** toggles Association **ON** and **OFF**. See [18.2.1 Leader Association](#)

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

**r** or **R** toggles between the arrow starting at the pick point (**free arrow**) or at the polygon centre (**fixed arrow**)

If Association is **ON** then the start of the Leader is associated with the centre of the string.

#### Step 2. Pick the hook point

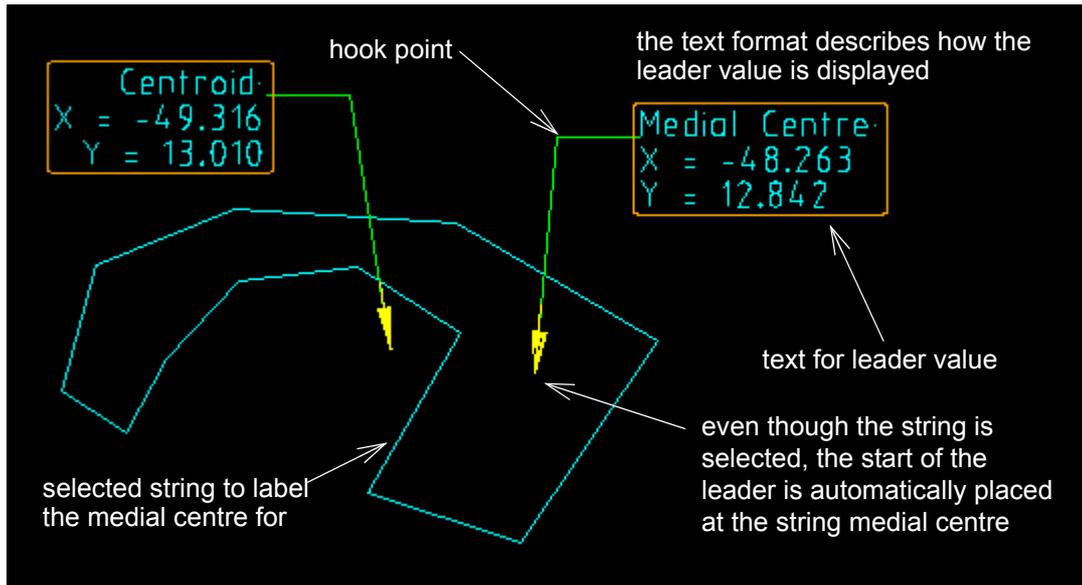
The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Information Leader* is then created.

The **Centre XY Information Leader** then starts again. That is, it goes back to **Step 1**.



**Note on Position of Leader Box With Respect to the Hook Point**

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the medial centre of the string then the leader box is drawn to the left of the hook point. If the hook point is to the right of the medial centre of the string then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.24 Trimesh Volume

The **Leader** choice **Trimesh volume** displays in the leader box, the volume of a selected closed trimesh.

A trimesh is selected (trimesh pick point) and a position for the hook and leader line is drawn from the trimesh pick point to a selected hook point, and then to the end of the hook itself.

Using the text format, the volume of the selected trimesh is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the trimesh to label with its volume

When **Trimesh Volume** is the choice, the following message is written to the screen message area

```
<Pick trimesh (s) tyle[default] (h)ook-angle[0°] > [picks][fast][Menu]
```

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

#### Step 2. Pick the hook point

The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Trimesh Volume Leader* is then created.

The **Trimesh Volume Leader** then starts again. That is, it goes back to **Step 1**.

#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

If the hook point is to the left of the medial centre of the string then the leader box is drawn to the left of the hook point. If the hook point is to the right of the medial centre of the string then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

### 18.2.5.25 Trimesh Surface Area

The **Leader** choice **Trimesh area** displays in the leader box, the surface area of a selected trimesh.

A trimesh is selected (trimesh pick point) and a position for the hook and leader line is drawn from the trimesh pick point to a selected hook point, and then to the end of the hook itself.

Using the text format, the volume of the selected trimesh is displayed at the end of the hook line.

The leader is added to the model given in the CAD toolbar.

#### Step 1. Pick the trimesh to label with its surface area

When **Trimesh area** is the choice, the following message is written to the screen message area

```
<Pick trimesh (s)tyle[default] (h)ook-angle[0°] > [picks][fast][Menu]
```

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**h** or **H** brings up the Hook Angle [18.2.2 Hook Angle](#)

#### Step 2. Pick the hook point

The following message is written to the screen message area.

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

The *Trimesh Area Leader* is then created.

The **Trimesh Area Leader** then starts again. That is, it goes back to **Step 1**.

#### Note on Position of Leader Box With Respect to the Hook Point

As well as determining the length of the leader line, the hook point also determines whether the leader box goes to the left or the right of the hook point.

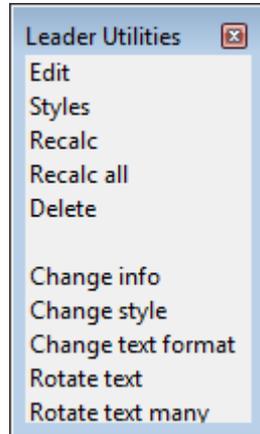
If the hook point is to the left of the medial centre of the string then the leader box is drawn to the left of the hook point. If the hook point is to the right of the medial centre of the string then the leader box is drawn to the right of the hook point.

For information on how the Leader object appears, see [18.2.6.2 Leader Styles](#).

## 18.2.6 Leader Utilities

Position of menu: **Cad =>Leader =>Utilities**

The **Leader Utilities** menu is:



edit a leader  
define styles for leaders

See

[18.2.6.1 Leader Edit](#)

[18.2.6.2 Leader Styles](#)

**Recalc** - select a leader to recalc.

**Recalc all** - recalcs all leaders.

**Delete** - select a leader to delete.

[18.2.6.3 Change Style of Leader](#)

[18.1.18.4 Change Text Format](#) (same option for Dimensions and Leaders)

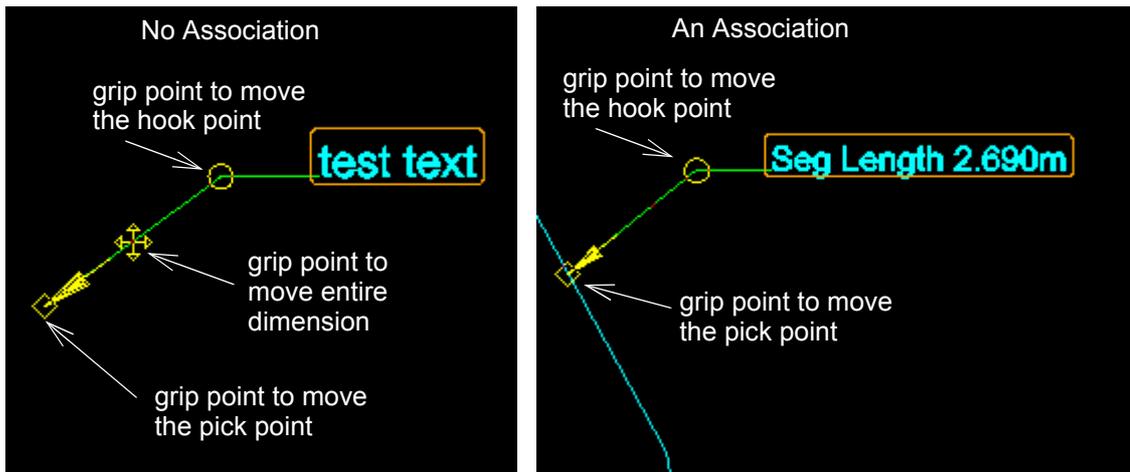
### 18.2.6.1 Leader Edit

Picking an existing Leader will bring up the Leader editor and the grip points for the pick point and the hook point are displayed.

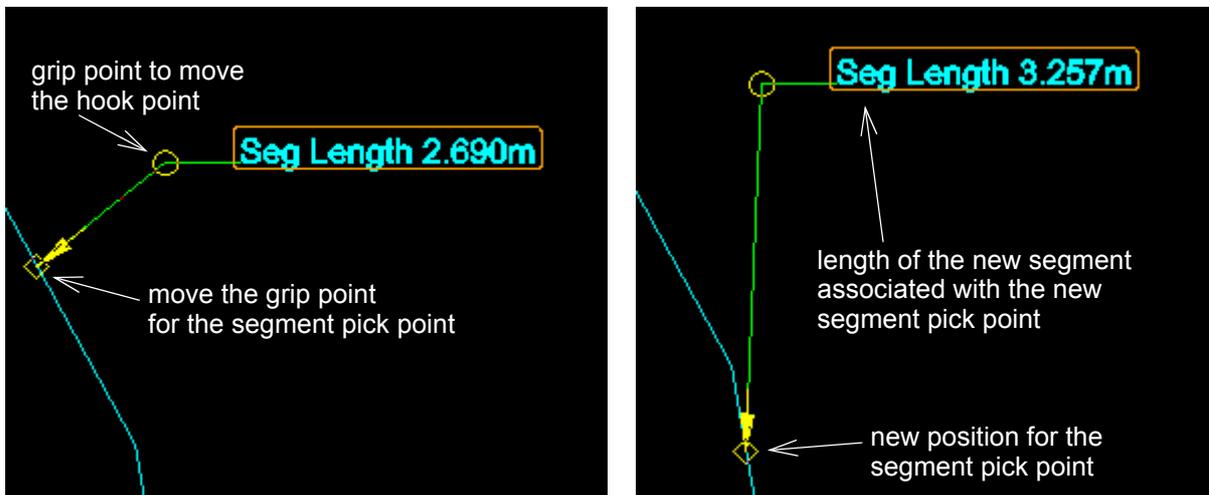
If there is **no Association** because either the Association is OFF, or Association is ON but there is nothing currently associated with the leader, then you won't get a grip point to move the entire leader because the leader is locked to its associated items.

A list of typed options is also displayed in the screen message area.

```
<[Pick grip point] (t)ext (s)tyle[default] (h)ook-angle[0°] (f)lip (d)elete a(n)other> [picks][fast][Menu]
```



Picking and accepting a grip point will move the grip point and when it is placed again, the association is recalced and the new value for the leader displayed.



For the typed options, typing

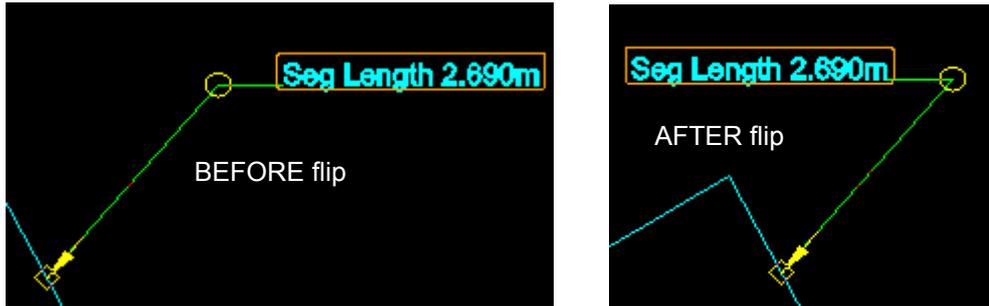
**t** or **T** brings up the leader text format. See [18.3 Text Format for Dimensions and Leaders](#).

**s** or **S** brings up the leader style list. [18.2.6.2 Leader Styles](#)

**f** or **F** to flip the side that the hook goes to.

**d** or **D** deletes the leader

**n** or **N** pick a new leader to edit.



Also

Pressing **<Esc>** will exit the editing of the current *leader* and ask for a new drafting element (Leader or Dimension) to be selected for editing.

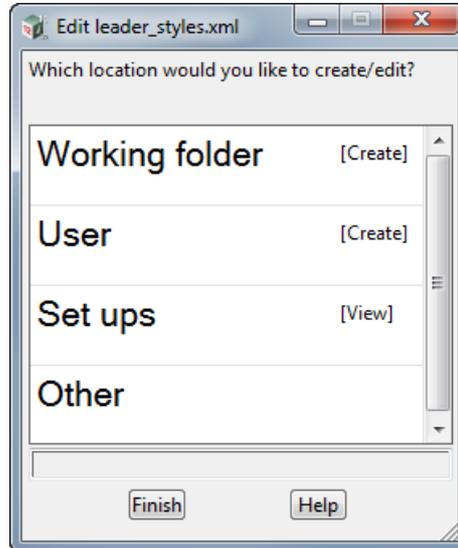
Similarly clicking **RB** and selecting **Cancel** from the **Pick Ops** menu will exit the editing of the current *leader* and ask for a new drafting element (Leader or Dimension) to be selected for editing.

When back in the Drafting Editor, pressing **<Esc>** or clicking **RB** and selecting **Cancel** from the **Pick Ops** menu, will exit the Drafting Editor

### 18.2.6.2 Leader Styles

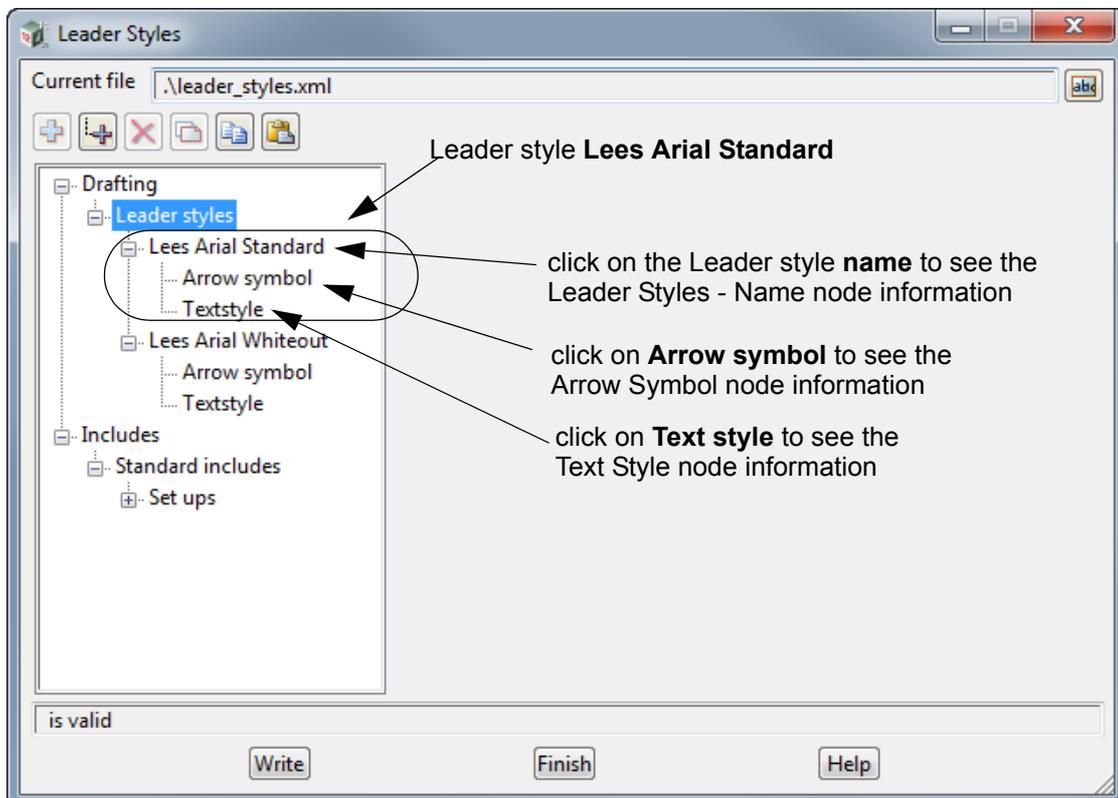
**Position of option on menu:** CAD =>Leader =>Styles

When you click on the option an **Edit leader\_styles.xml** panel is brought up and the panel shows the standard areas for looking for the leader\_styles.XML files - Working folder, Customer (User), User, Set Ups and Other.



For information about where how to find and create dimension\_styles.xml files, see [18.5 Style XML Files for Dimensions, Leaders and Tables](#).

Once a leader\_styles.xml file has been opened, the **Leader Styles** panel is displayed.



See [18.2.6.2.1 Leader Styles - Name Node](#)

[18.2.6.2.2 Leader Styles - Arrow Symbol Node](#)

[18.2.6.2.3 Leader Styles - Hook Symbol Tab](#)

[18.2.6.2.4 Leader Styles - Textstyle Node](#)

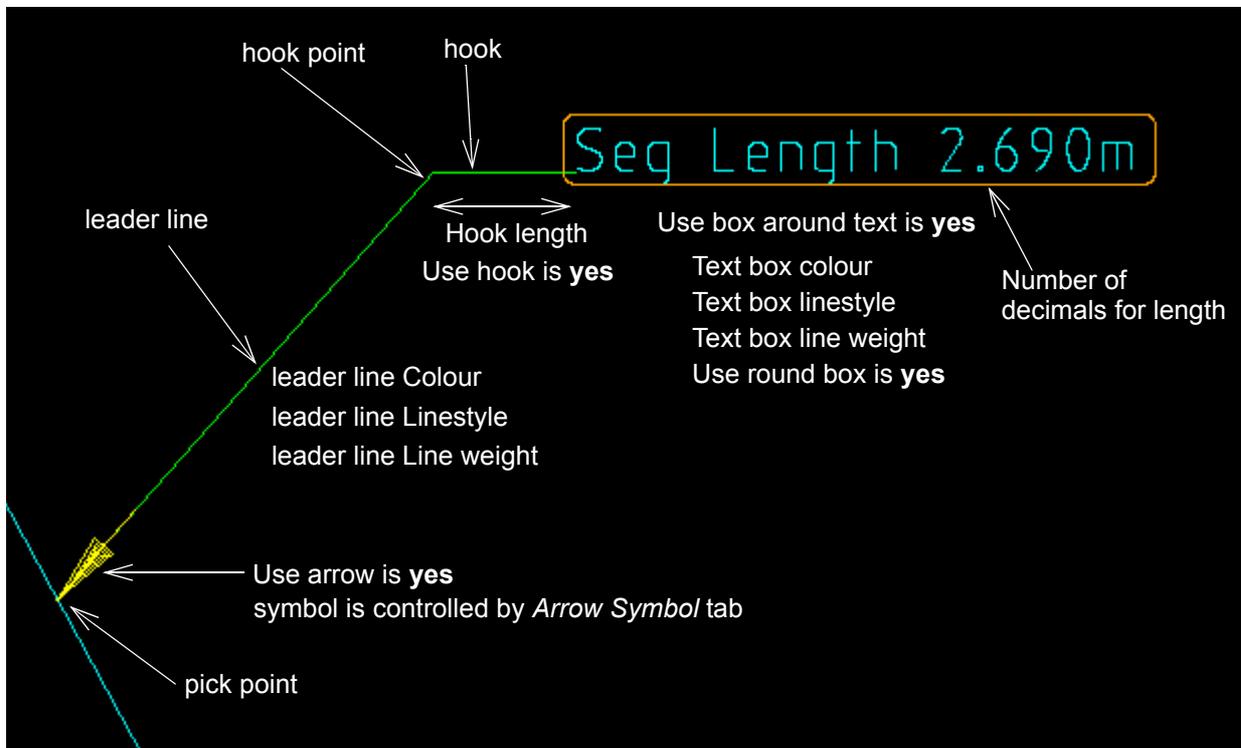
### 18.2.6.2.1 Leader Styles - Name Node

This is on the right hand side of the panel when you click on the **Leader style** name.

Name	Paper	abc
Group	no hook	abc
Colour	green	■
Linestyle	THIN	abc
Lineweight	1	1
Use arrow	yes	▼
Use hook	yes	▼
Hook length	0.5	1
Use box around text	yes	▼
Use rounded box	yes	▼
Text box colour	orange	■
Text box linestyle	1	1
Text box lineweight	0	1
Number of decimals for length	3	123
Number of decimals for area	2	123
Use dynamic size	no	▼
Hook angle	0	1
Angle format style	show_angle	▼
Number of decimal for angle	2	123
Number of decimals for volume	2	123

**Hook length** has same units as Textstyle. So if the Textstyle is paper, then **Hook length** is in millimetres.

these settings are not currently used



If the **Type** from the **Textstyle** tab is **not world**, then the following are not used.

Use text box - should be *Use box around text* like dimension.

Text box rounded - should be *Use rounded box* like dimension?.

Text box colour

Text box linestyle

Text box line weight

The following items that are at the bottom of the tab have not yet been implemented.

Hook angle

Use dynamic size

Number of decimals for volumes

Angle format style

### 18.2.6.2.2 Leader Styles - Arrow Symbol Node

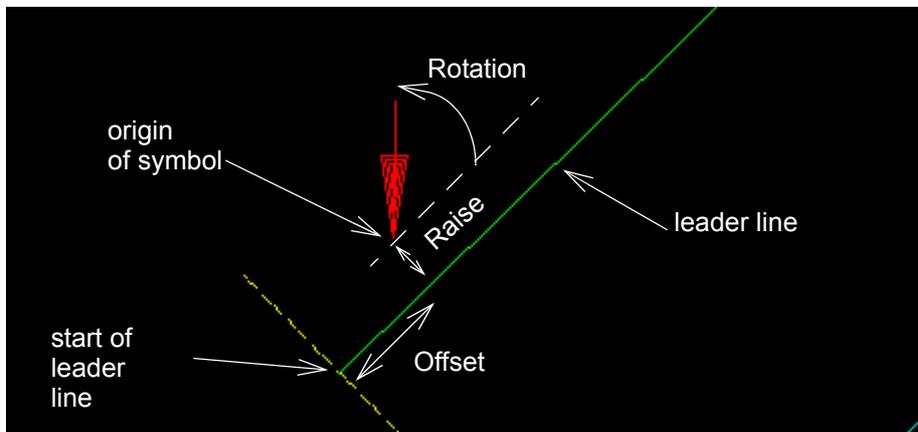
This is on the right hand side of the panel when you click on the **Arrow symbol** node.

Style	Arrow	
Colour	red	
Size	0.5	
Rotation	45	
Offset	0.2	
Raise	0.5	

The origin of the symbol is displaced from the **start** of the leader line by the **Offset** and **Raise** values.

With a **Rotation** value of zero, the symbol is given the same rotation as the leader line.

For a non zero **Rotation**, the value is **added** to the angle of the leader line.



### 18.2.6.2.3 Leader Styles - Hook Symbol Tab

Not implemented at this stage.

Style	ARROW	
Colour	yellow	
Size	0.5	
Rotation	45	
Offset	0.2	
Raise	0.5	

### 18.2.6.2.4 Leader Styles - Textstyle Node

This is on the right hand side of the panel when you click on the **Textstyle** node.

Textstyle	Arial	T
Name		abc
Type	paper	▼
Size	0.2	⊞
Angle	0°	⊞
X-factor	1	⊞
Slant	0	⊞
Offset	0	⊞
Raise	0	⊞
Colour	cyan	■
Whiteout	no colour	
Border	no colour	
Border type		▼
Justify	bottom-left	▼
TTF Underline	no	▼
TTF Strikeout	no	▼
TTF Italic	no	▼
TTF Outline	no	▼
TTF Weight	400	⊞

This is expected to be modified shortly so that it uses a *Textstyle Data* and all the features from *Textstyle Data* are automatically available. For the moment with paper text, only **Whiteout** is supported.

When the *Textstyle Data* is implemented, it can have its own *Border type* but there is no linestyle for the border in *Textstyle Data*. The only way to get a border with a linestyle is to use World text and set the border parameters in the *General* tab ([18.2.6.2.1 Leader Styles - Name Node](#)).

If **Type** is not **world**, then **Don't clip dimension line for text** is set to **no** and these items from the *General* tab are not used.

- Use box around text
- Use rounded box
- Text box colour
- Text box linestyle

See [18.2.6.2.1 Leader Styles - Name Node](#)

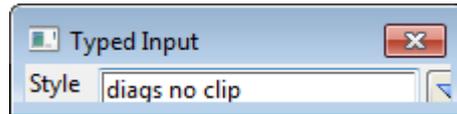
### 18.2.6.3 Change Style of Leader

Click on **Change style of style** and the following message is written to the screen message area

<Pick drafting element to apply (s)tyle[diags no clip]> [picks][fast][Menu]

Pick a leader and its style will be changed to that shown in the square brackets ([]). Then pick another leader to change its style.

Typing **s** or **S** brings up the **Style Typed Input** box to select a new leaders style.



The new style is picked from the pop up list, or typed into the box and <Enter> pressed.

The box is then removed and the message again written to the screen message area.

All subsequent selected leaders will have their styles changed to this new style until either the option is terminated or a new leader style is selected

To terminate the option, press <Esc>, select **Cancel** from the **Pick Ops** menu, or select another CAD option.

## 18.3 Text Format for Dimensions and Leaders

For *Dimensions* and *Leaders*, there is one, and sometimes more than one, value that is to be displayed in the *Dimension* or *Leader* as text. For example, the length of a segment or the x and y coordinates of a point.

The user rarely want to display all the different dimension and leader values in the same way. For example, some values may be displayed in metres, others in millimetres. Or you may want to display both metres and yards.

In 12d Model, there are special **Text Formats** to allow the user to tailor how the values of the *Dimensions* and *Leaders* are displayed.

So a **Text Format** is used to describe how a single value for a dimension or leader is displayed.

In general it has the form

$$pre\_text\mathbf{Expression}post\_text$$

where **Expression** can be either

- (a) a set of angle brackets enclosing a *value\_format* which describes how the value is modified and how many decimal places are used in the final text display of the value

$\langle value\_format \rangle$

or

- (b) an expression involving an **IF** statement (known as a **dynamic text format**)

The full structure of the **Text Format**, the *value\_format* and dynamic text format is described in the following sections:

[18.3.1 Angle Brackets <>](#)

[18.3.2 Pre and Post Text](#)

[18.3.3 Units Factor](#)

[18.3.4 Round Off](#)

[18.3.5 Number of Decimals](#)

[18.3.6 Dynamic Text Format](#)

[18.3.7 Multiple Text Formats](#)

[18.3.8 Typing t or T to Change the Text Format](#)

[18.3.9 Defaults File for Text Formats](#)

## 18.3.1 Angle Brackets <>

In the Text Formats, angle brackets <> are used to indicate the position of the value of the *Dimension* or *Leader*.

There can be special value formatting instructions inside the angle brackets to indicate if and how the value is to be modified, and how many decimals to use when finally displayed as text.

<value\_format>

What is available for *value\_format* will be described in the following sections.

In some cases, the angle brackets may not appear at all in the Text Format and then the text does not include the actual value itself.

If the *Dimension* or *Leader* has only one value, then just <> is required.

If the *Dimension* or *Leader* has more than one value, then extra commands are needed to indicate which value is being talked about in the Text Format.

<x> is used for the x value of a coordinate

<y> is used for the y value of a coordinate

<z> is used for the z value of a coordinate

## 18.3.2 Pre and Post Text

The text displayed for a dimension or leader value can have user defined *pre* and *post* text around the value and that is indicated by the expression

$$pre\_text<value\_format>post\_text$$

where **pre\_text** is written before the dimension or leader value, **post\_text** is written after the value.

Typing \n into **pre\_text** or **post\_text** will be interpreted as a new line.

The angle brackets <> indicate where you want the value in relation to the **pre\_text** and **post\_text**. The angle brackets may even be missing.

For example, to show just the word “varies”, use the following with no angle brackets.

varies

## 18.3.3 Units Factor

The value to be displayed can be multiplied by a **units\_factor**. The *units\_factor* is written inside the angle brackets <>. That is

$$<units\_factor>$$

So the expression

$$pre\_text<units\_factor>post\_text$$

means that **pre\_text** is written before the dimension or leader value, **post\_text** is written after the value, <> indicates that you want the value and the value is multiplied by the **units\_factor** before being displayed.

If **units\_factor** is left blank then it is defaulted to 1.

If <**units\_factor**> is left out, then there is no value in the dimension or leader text but you still get the **pre\_text** and **post\_text**.

### Note

*pre\_text<units\_factor>post\_text* is considered to be **one** block of text. So raise, offset, justification etc apply to the whole block, not to the *pre\_text* or *post\_text* individually.

As an example of *pre\_text<units\_factor>post\_text*, you can show millimetres instead of metres by using:

$$<1000>$$

or, if you want the text “ mm” after the value

$$<1000> mm$$

## 18.3.4 Round Off

Inside the angle brackets <>, the type of Rounding Off can be specified by a key character **O** (down), **M** (middle) or **U** (up) followed immediately (with no spaces) by the **rounding\_unit**.

- O**rounding\_unit means to round down to the closest rounding\_unit
- M**rounding\_unit means to round to the closest rounding\_unit
- U**rounding\_unit means to round up to the closest rounding\_unit

For example

- O**1 means that 10.2 would round down to 10.
- O**0.1 means that 10.2 would round down to 10.2
- D**0.25 means that 10.2 would round up to 10.
- U**0.25 means that 10.2 would round up to 10.25
- M**0.25 means that 10.2 would round up to 10.25 rather than down to 10.

**Note:** the *units\_factor* is applied first and then the *Round Off*.

## 18.3.5 Number of Decimals

Inside the angle brackets <>, you can give the number of decimal places to use when writing out the real value by using the key character **D** followed immediately (with no spaces) by the **number of decimal places**.

- D***n* where **n** is the number of decimal places.  
*pre\_text*<*units\_factor***D***n*>*post\_text*

The *units\_factor* is applied first and then the number of decimal places used on the resulting value.

For example

<1000**D**1>

says to multiply the real by 1000 and only use one decimal place in the text for the value.

**Note:** the *units\_factor* is applied first, followed by the *Round Off*, and finally the *number of decimal places* used in the text to display the resulting value.

So <100**D**2**U**0.25> is the same as <100**U**0.25**D**2> - multiply by 100, round Up using the *rounding\_unit* of 0.25, and then writing the result with two decimal places.

## 18.3.6 Dynamic Text Format

It is also possible to change the format for a value depending on the size of the value by using an **Expression** with an **IF** test instead of the simple angle brackets `< >`.

The **Expression** has key words `_IF_`, `_THEN_`, `_ELSE_` and `_ELSEIF_` and can be

```
_IF_ op1 comparison_value_1 _THEN_ single_value_format_1 _ELSEIF_ op2
comparison_value_2 _THEN_ single_value_format_2 _ELSE_IF_ op3 comparison_value_3
_THEN_ single_value_format_3..._ELSE_ single_value_format_n
```

where **op1** and **op2** are one of

```
=          for equals
>          for greater than
>=         for greater than or equal to
<          for less than
<=         for less than or equal to
```

and

`single_value_format_n` is a **Text Format** for a single value and **not** an Expression.

The comparison is always being made between the *dimension/leader value* and the *comparison\_value* so we have left out the *dimension/leader value*.

That is, `_IF_ >comparison_value` means *if (dimension value is greater than comparison\_value)*.

For example if you wanted to display kilometres to three decimal place, for anything greater than or equal to 1,000, metres to three decimal places for values greater or equal to 1, otherwise millimetres with no decimal places, then use

```
_IF_ >=1000 _THEN_ <0.001D3> km _ELSEIF_ >=1 _THEN_ <D3> m _ELSE_ <1000D0>mm
```

### Important Notes

1. The *Dynamic Text Format* always evaluates to only one piece of text.
2. The underscore is part of `_IF_`, `_THEN_`, `_ELSEIF_` and `_ELSE_`.
3. `_IF_ op comparison_value` must be followed by a `_THEN_ single_value_format`.
4. `_ELSEIF_ op comparison_value` must be followed by a `_THEN_ single_value_format`.
5. It is compulsory to have an `_ELSE_` at the end of the **Expression**.
6. An **Expression** is only for `<>` values and **not** `<x>`, `<y>` etc.

## 18.3.7 Multiple Text Formats

You can have just the one Text Format or multiple text formats in the dimension or leader.

That is, you can have

```
pre_text<value_format_1>text_1<value_format_2>...text_n<value_format_n>
```

where **pre\_text** is written before the value with `value_format_1`, **text\_n** is written after the n'th `<>` and the value in the n'th `<>` has the format `value_format_n`.

For example, to show metres and International feet, use

```
<> m or <3.28083895> ft
```

or to show metres and US Survey feet, use

```
<> m or <3.280833333> ft
```

### 18.3.8 Typing t or T to Change the Text Format

When using many of the **Dimension** or **Leader** commands, a list of typed options is often displayed in the screen message area and this may include **forma(t)** or **customise (t)ext**.

#### example for Dimensions

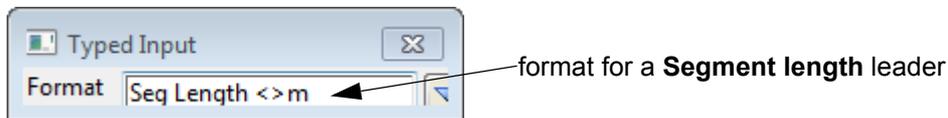
```
<[Pick start point] (a)ssociative (s)tyle[default] forma(t) [(z)ero offset OFF] > [picks][fast][Menu]
```

#### for Information Leaders

```
<Pick hook point or customise (t)ext> [picks][fast][Menu]
```

When this is the case, typing **t** or **T** is for changing the Text Format for the dimension or leader. The default value is the last one used.

After typing **t** or **T**, the **Format Typed Input** box is displayed showing the current text format of the selected dimension or leader.



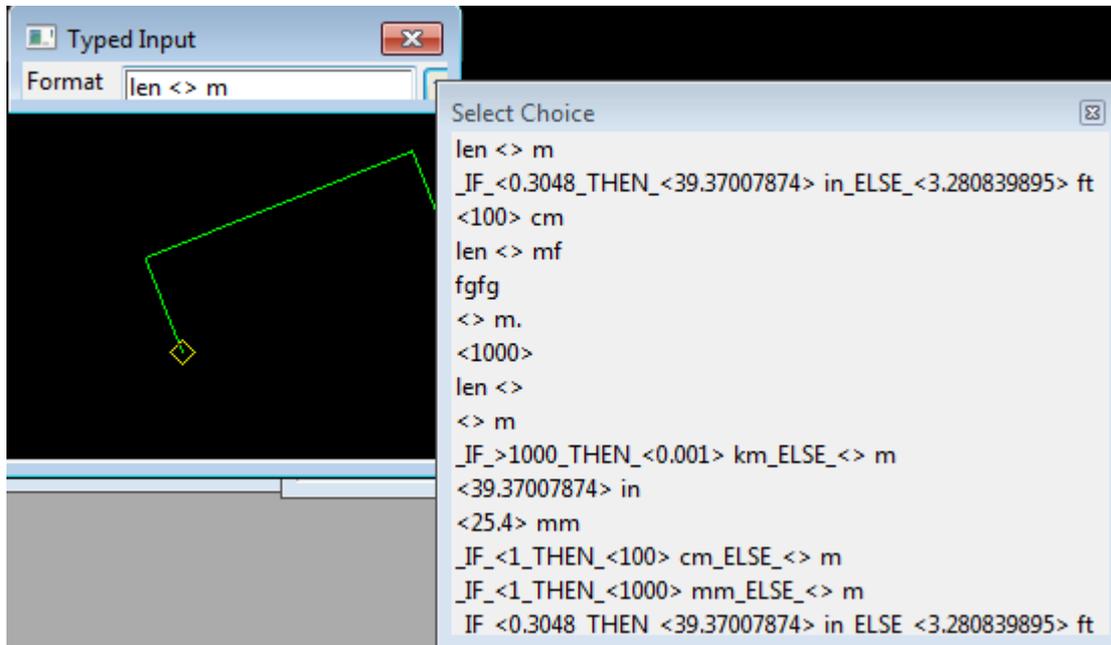
Clicking on the Choice icon will bring up a list of *Text Formats* for the type of *dimension* or *leader* being worked on. This includes the **last six** used which is useful when using the same format multiple times. See [18.3.9 Defaults File for Text Formats](#).

The new *Text Format* is picked from the pop up list, or typed into the box and **<Enter>** pressed. The **Format Typed Input** box is then removed.

## 18.3.9 Defaults File for Text Formats

There is an XML file called *draft\_texts.xml* which contains up to fifteen text formats for **each** *Dimension* type, and up to fifteen text formats for **each** type of *Information Leader*.

When you type **t** or **T** after seeing **(t)ext** included in the message in the screen message area, a **Format Typed Input** box is placed on the screen. The pop on the box up brings up a list of up to fifteen entries which come from the appropriate section of the *draft\_texts.xml* file for the dimension or leader type being created or edited.



The first **six** of the fifteen in the pop up are the last six ones used in the current session of **12d Model**, and the rest come from the *drafts\_texts.xml* file.

So in a new session, all the entries in the pop up come from the *drafts\_texts.xml* file. Each time one is selected from the list, or a new text format typed in, the selected or typed in text format is placed on the top of the list. If there is already six new entries on the top of the list, the sixth one is deleted and the first five are shuffled down.

As each of the six new entries are added to the top of the list, the bottom six that have come from the *draft\_texts.xml* file are dropped off the list.

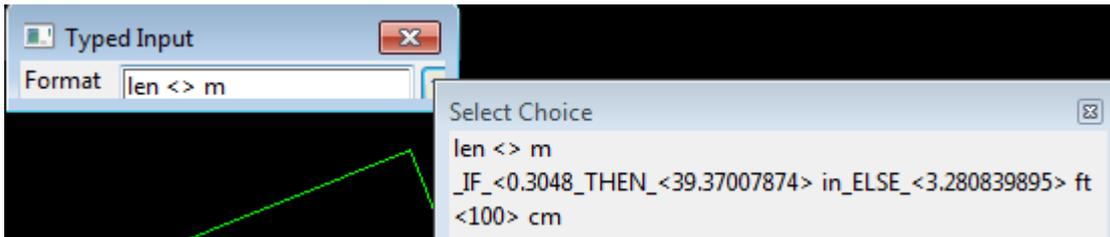
The *draft\_texts.xml* file is read each time you open a **12d Model** project.

### 18.3.9.1 An Excerpt from a draft\_text.xml File

The *draft\_text.xml* file consists of a <meta\_data> block at the top and then a <drafting> block with <Leader\_Text> and <Dimension\_Text> sections.

**Note** that because < and > are part of the XML syntax, in an XML file, less than (<) is written as &lt;; and greater than (>) is written as &gt;.

An example of part of the *Dimension\_Text* and *Leader\_Text* sections of the *drafts\_texts.xml* file, which includes the Aligned\_Type dimension text formats for the first three entries in the pop up list.



is:

#### <Dimension\_Text>

```
<Aligned_Type0>len &lt;&gt; m</Aligned_Type0>
<Angular_Type0>&lt;&gt;</Angular_Type0>
<Arc_Ang_Type0>&lt;&gt;</Arc_Ang_Type0>
<Aligned_Type1>_IF_&lt;0.3048_THEN_&lt;39.37007874&gt; in_ELSE_&lt;3.280839895&gt;
ft</Aligned_Type1>
<Angular_Type1>&lt;&gt;</Angular_Type1>
<Arc_Ang_Type1>&lt;&gt;</Arc_Ang_Type1>
<Aligned_Type2>&lt;100&gt; cm</Aligned_Type2>
```

#### </Dimension\_Text>

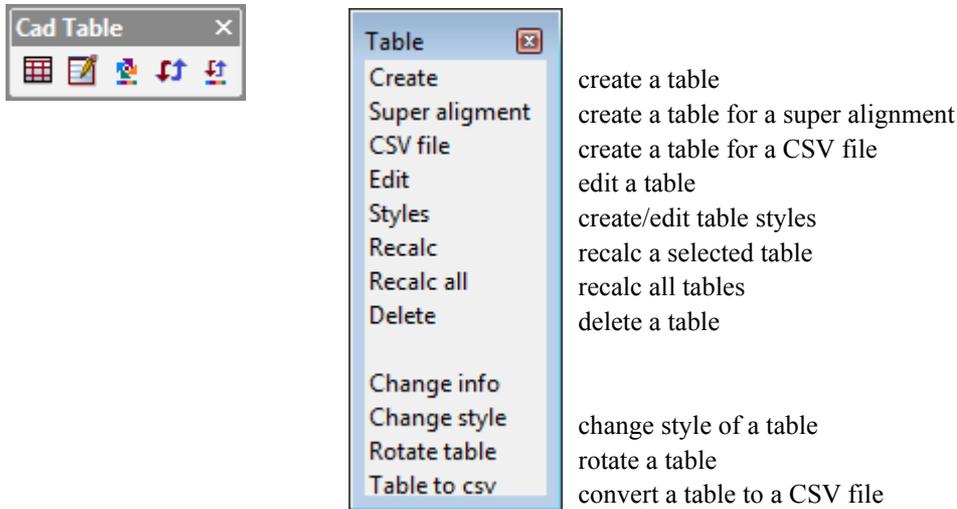
#### <Leader\_Text>

```
<Unknown_Type0>sadfsdfa\nLine2</Unknown_Type0>
<Area_Type0>Area &lt;&gt;</Area_Type0>
<String_Length_3d_Type0>Length 3d &lt;&gt;m</String_Length_3d_Type0>
<Unknown_Type1>sadfsdfa</Unknown_Type1>
<Area_Type1>Area &lt;&gt;</Area_Type1>
<String_Length_3d_Type1>Length 3d &lt;&gt;m</String_Length_3d_Type1>
<Unknown_Type2>sadf</Unknown_Type2>
<Area_Type2>Area &lt;&gt;</Area_Type2>
<String_Length_3d_Type2>Length 3d &lt;&gt;m</String_Length_3d_Type2>
```

#### </Leader\_Text>

## 18.4 CAD Table

There is a new **CAD Table** toolbar and **CAD =>Table** menu.



The **CAD Table** options create and edit tables.

There are options to create some tables from super alignments and they are associated with the super alignment so they can automatically update with changes in the super alignment.

See

[18.4.1 Create Table](#)

[18.4.2 Create Table from a Super Alignment](#)

[18.4.3 Create Table from a CSV File](#)

[18.4.4 Table Edit](#)

[18.4.5 Table Styles](#)

**Recalc** - select a table to recalc.

**Recalc all** - recalcs all tables.

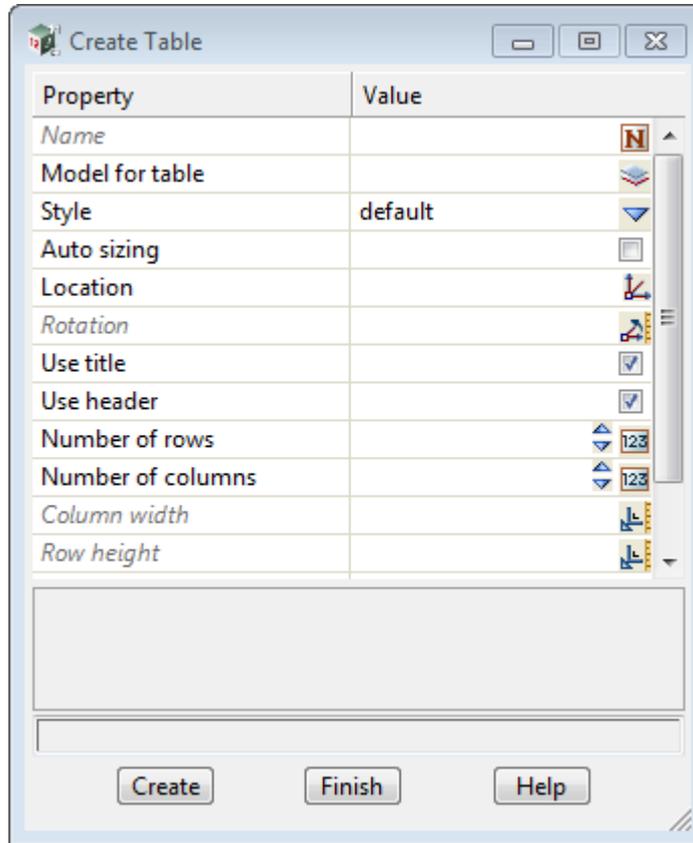
**Delete** - select a table to delete.

[18.4.6 Table to CSV](#)

## 18.4.1 Create Table

Position of option on menu: CAD =>Table =>Create

Selecting Create brings up the Create Table panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Name</b> <i>the name for the table. This must not be blank.</i>	name box		available names
<b>Model for table</b> <i>the model for the table.</i>	model box		available models
<b>Style</b> <i>the table style to use for this table.</i>	table style box		available table styles
<b>Auto sizing</b> <i>if <b>ticked</b>, the table automatically resizes the column widths and heights to fit the text into the columns. If <b>not ticked</b>, the table will not resize the column widths and heights to fit the text into the column.</i>	tick box	not ticked	
<b>Location</b> <i>the (x,y) coordinates of the <b>top left hand corner</b> of the table.</i>	x y box	0	
<b>Rotation</b> <i>the rotation of the table - entered in hp notation. It is measured in a counter clockwise direction from the positive x axis.</i>	angle box	0	
<b>Use title</b> <i>if <b>ticked</b>, the table has a title as the first row and it only consists of one column which goes from the left side of the table to the right side of the table. The textstyle information for text in the title is given in the</i>	tick box	ticked	

**Title style** node of the *Drafting Styles for Tables*.

If **not ticked**, the table does not have a title at the top.

**Use header** tick box  **ticked**

if **ticked**, each column has as its first entry a header. The textstyle information for text in the row of headers is given in the **Header style** node of the *Drafting Styles for Tables*

If **not ticked**, the table does not have a title at the top.

**Number of rows** integer box

the number of rows in the table.

**Number of columns** integer box

the number of columns in the table.

**Column width** real box

if **Auto sizing** is **no**, the width of each column.

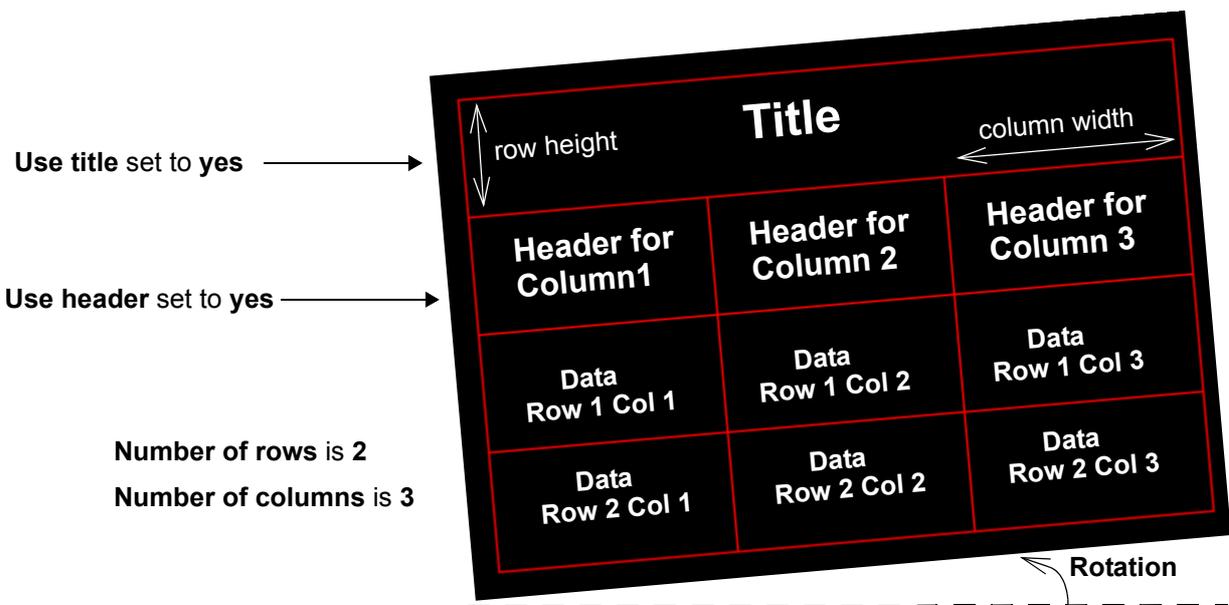
**Row height** real box

if **Auto sizing** is **no**, the height of each column.

**Buttons at bottom**

**Create** button

create the table with the properties in this panel at the location given in **Location**.

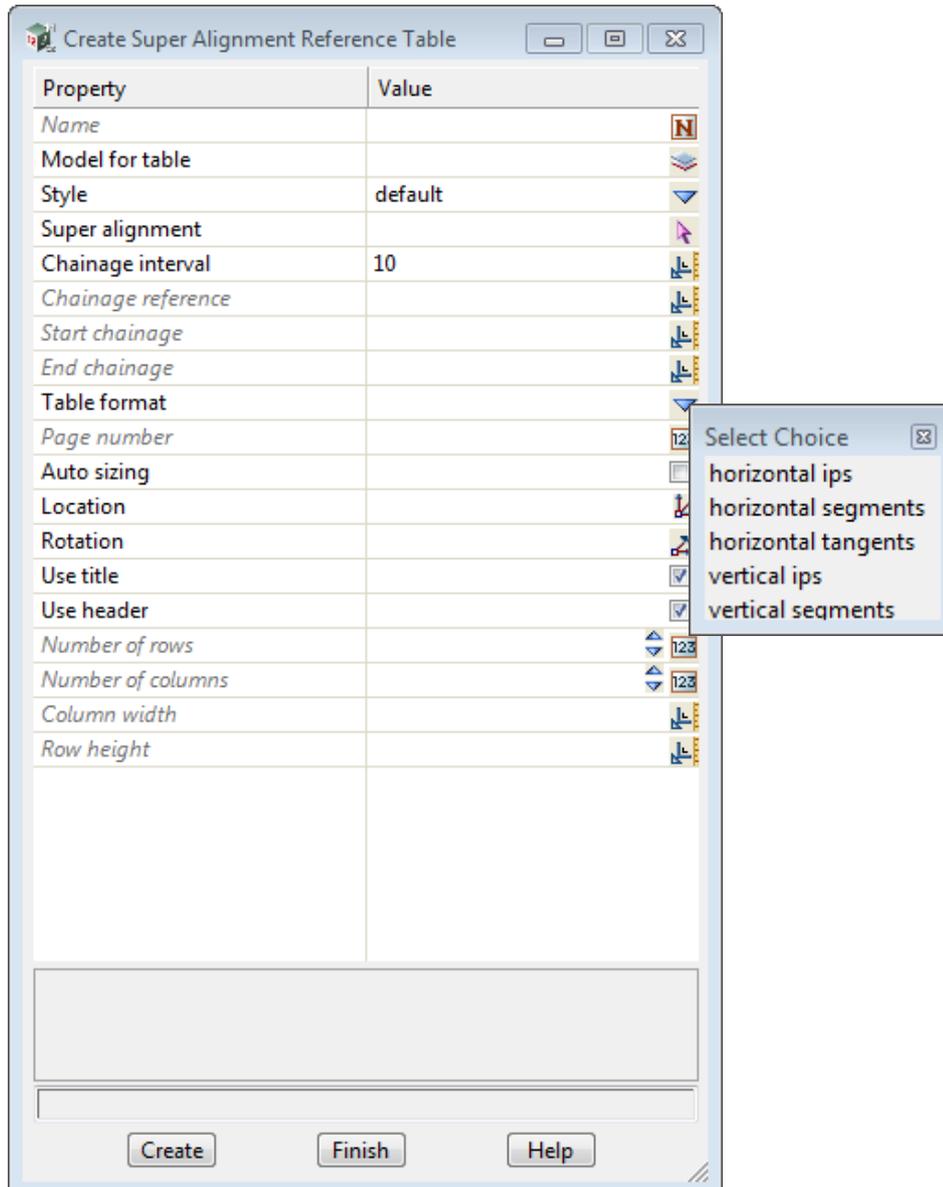


## 18.4.2 Create Table from a Super Alignment

**Position of option on menu:** CAD =>Table =>Super alignment

This table option is for creating a table of information about a selected **super alignment**. The table is associated with the super alignment so that the table can be updated when the super alignment is modified.

Selecting **Super alignment** brings up the **Create Super Alignment Reference Table** panel.



The fields and buttons used in this panel have the following functions.

Field	Description	Type	Defaults	Pop-Up
<b>Name</b>	<i>the name for the table. This must not be blank.</i>	name box		available names
<b>Model for table</b>	<i>the model for the table.</i>	model box		available models
<b>Style</b>	<i>the table style to use for this table.</i>	table style box		available table styles

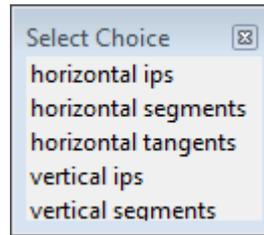
**Super alignment** string select box  
*select the super alignment to provide the information to generate the table.*

**Chainage interval** real box  
*the interval to use if reporting information at chainage intervals down the super alignment.*

**Chainage reference** real box  
*the intervals for reporting are incremented about the Chainage reference value.*

**Start/End chainage** choice box  
*if **not blank**, the chainage to use for the Start/End chainages reported in the table.  
 If **blank**, the Start/End chainage of the super alignment is used.*

**Table format** choice box



*if **horizontal ips**  
 If **horizontal segments**,  
 If **horizontal tangents**,  
 If **vertical ips**,  
 If **vertical segments**,*

**Page number** integer box

**Auto sizing** tick box not ticked  
*if **ticked**, the table automatically resizes the column widths and heights to fit the text into the columns.  
 If **not ticked**, the table will not resize the column widths and heights to fit the text into the column.*

**Location** x y box 0  
*the (x,y) coordinates of the **top left hand corner** of the table.*

**Rotation** angle box 0  
*the rotation of the table - entered in hp notation. It is measured in a counter clockwise direction from the positive x axis.*

**Use title** tick box ticked  
*if **ticked**, the table has a title as the first row and it only consists of one column which goes from the left side of the table to the right side of the table. The textstyle information for text in the title is given in the **Title style** node of the Drafting Styles for Tables.  
 If **not ticked**, the table does not have a title at the top.*

**Use header** tick box ticked  
*if **ticked**, each column has as its first entry a header. The textstyle information for text in the row of headers is given in the **Header style** node of the Drafting Styles for Tables  
 If **not ticked**, the table does not have a title at the top.*

**Number of rows** integer box  
*the number of rows in the table.  
 Not used - the Table format and super alignment determines how many rows there are.*

**Number of columns** integer box  
*the number of columns in the table.*

*Not used - the Table format and super alignment determines how many columns there are.*

**Column width**                      real box

*if Auto sizing is no, the width of each column.*

**Row height**                      real box

*if Auto sizing is no, the height of each column.*

**Buttons at bottom**

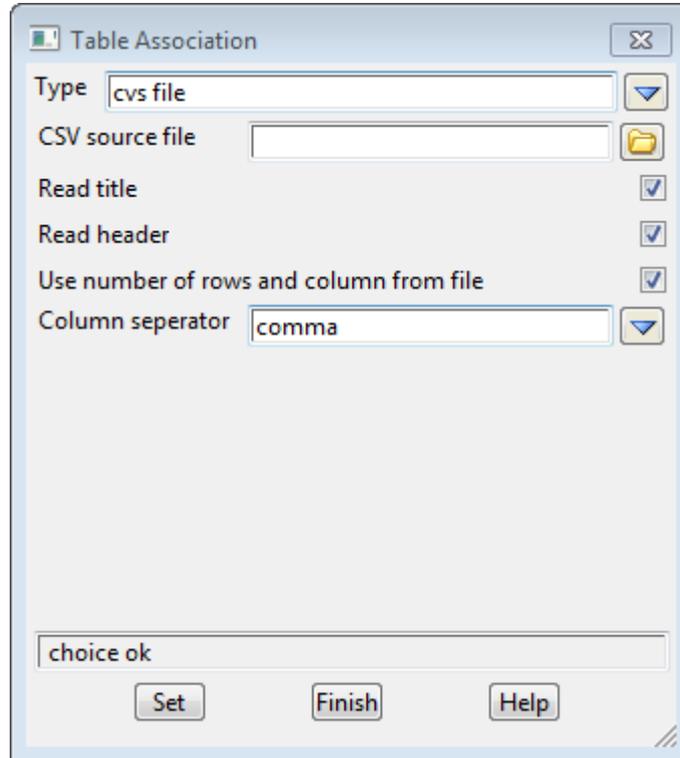
**Create**                              button

*create the table with the properties in this panel at the location given in **Location**.*

### 18.4.3 Create Table from a CSV File

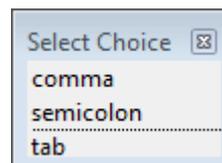
This table option is for creating a table of information from the information in a CSV file. The table is associated with the CSV file so that the table can be updated when the CSV file is modified.

Selecting **CSV file** brings up the **Create CSV File Reference Table** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Name</b> <i>the name for the table. This must not be blank.</i>	name box		available names
<b>Model for table</b> <i>the model for the table.</i>	model box		available models
<b>Style</b> <i>the table style to use for this table.</i>	table style box		available table styles
<b>CSV source file</b> <i>name of the CSV file to associate with the table.</i>	file box		
<b>Column separator</b>	choice box		



*if **comma**, in the CSV file a comma (,) is used the separate the information for each column.*  
*If **semicolon**, in the CSV file a semicolon (;) is used the separate the information for each column.*  
*If **tab**, in the CSV file a tab is used the separate the information for each column.*

**Auto sizing**                      tick box                      not ticked  
*if **ticked**, the table automatically resizes the column widths and heights to fit the text into the columns.  
If **not ticked**, the table will not resize the column widths and heights to fit the text into the column.*

**Location**                      x y box                      0  
*the (x,y) coordinates of the **top left hand corner** of the table.*

**Rotation**                      angle box                      0  
*the rotation of the table - entered in hp notation. It is measured in a counter clockwise direction from the positive x axis.*

**Use title**                      tick box  
*if **ticked** then the first line of the CSV file is taken as the titles for each column in the table. The textstyle information for text in the title is given in the **Title style** node of the Drafting Styles for Tables.  
If **not ticked**, the table does not have a title.*

**Use header**                      tick box  
*if **ticked** then the next line of the CSV file is taken to be the Headers for each column of the table. The textstyle information for text in the row of headers is given in the **Header style** node of the Drafting Styles for Tables  
If **not ticked**, the table does not have a header.*

**Number of rows**                      integer box  
*the number of rows in the table.  
Not used - the Table format and super alignment determines how many rows there are.*

**Number of columns**                      integer box  
*the number of columns in the table.  
Not used - the Table format and super alignment determines how many columns there are.*

**Column width**                      real box  
*if **Auto sizing** is **no**, the width of each column.*

**Row height**                      real box  
*if **Auto sizing** is **no**, the height of each column.*

### Buttons at bottom

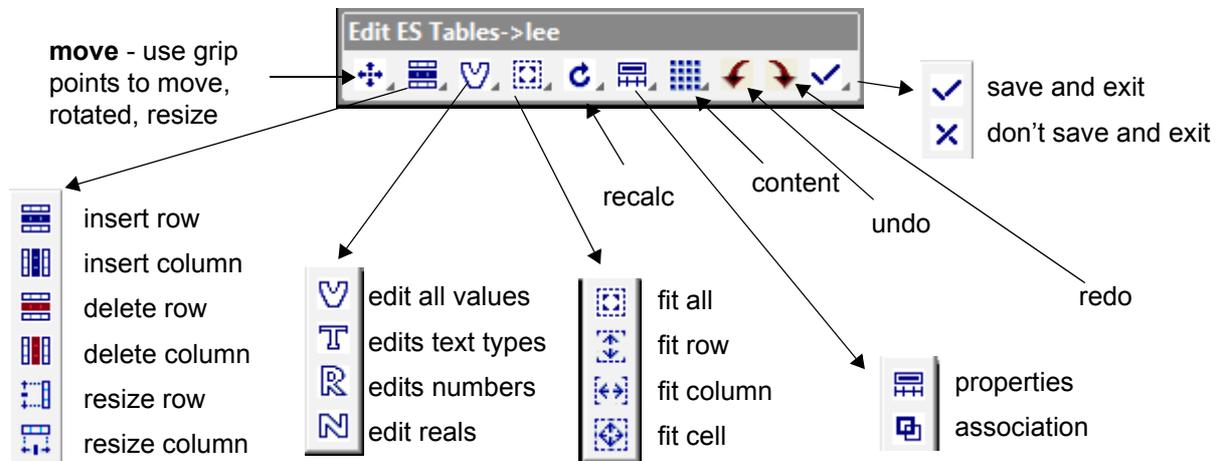
**Create**                      button  
*create the table with the properties in this panel at the location given in **Location**.*

### Important Notes

1. When a **Recalc** is done, the CSV file is reread and the information in the table updated.
2. **Auto size** in the table Properties must be ticked on for the new table to be automatically resized.
3. If you don't want the CSV file to be reread on a **Recalc**, you need to set **Type** back to **no reference** in the **Table Association** panel.

## 18.4.4 Table Edit

Picking an existing table will bring up the **Table** editor, and at the same time the grip points and point are displayed on the plan views that the table is on (see [18.4.4.1 Move](#)).



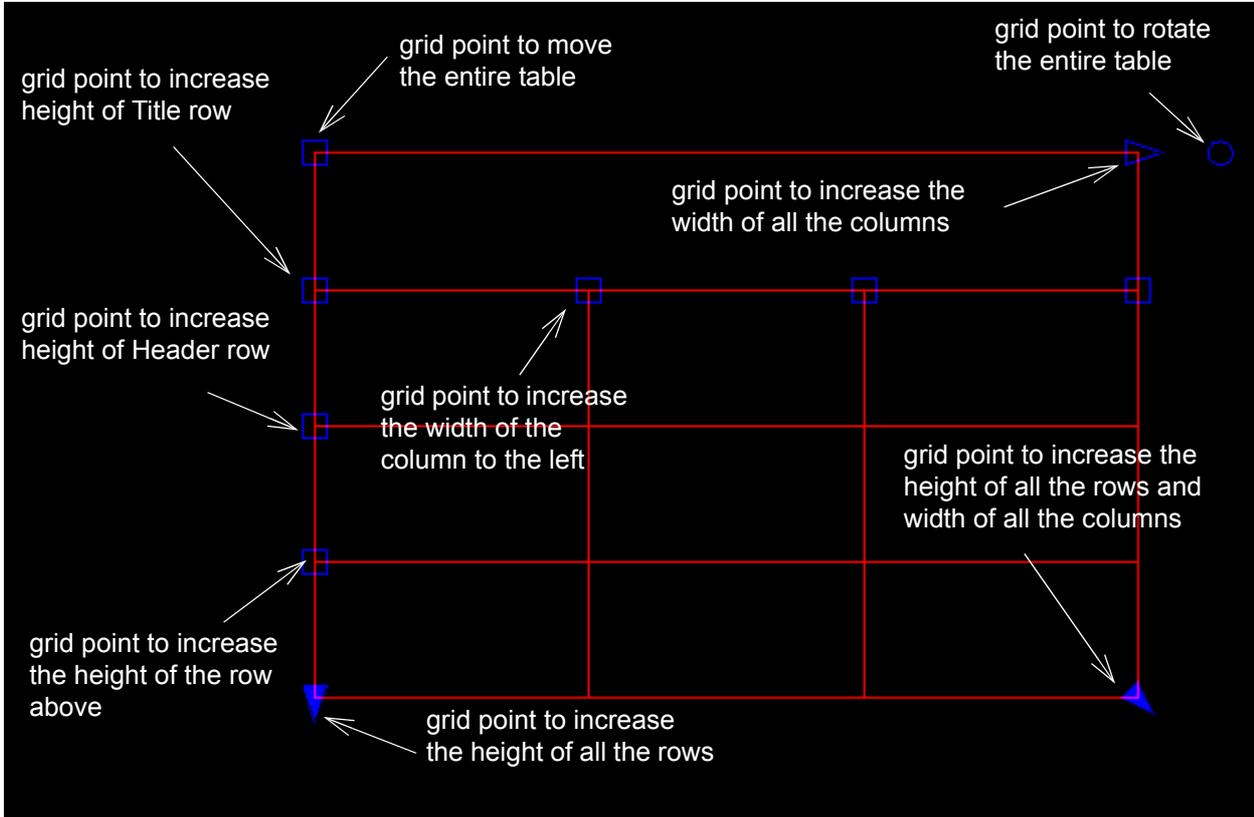
See

- [18.4.4.1 Move](#)
- [18.4.4.2 Column and Row Edits](#)
- [18.4.4.3 Text and Values Edit](#)
- [18.4.4.4.1 Fit All](#)
- [18.4.4.5 Recalc](#)
- [18.4.4.6 Properties and Associations](#)
- [18.4.4.7 Content](#)

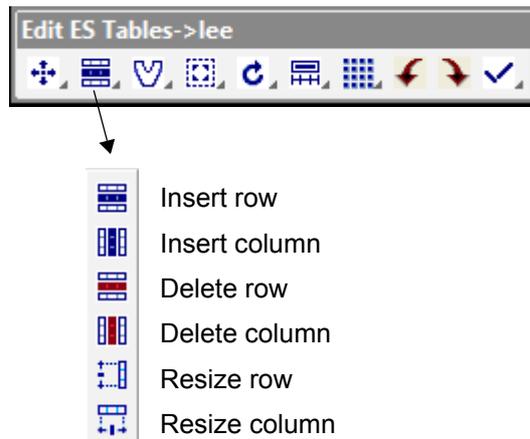
### 18.4.4.1 Move

When the **Table Editor** is running, the grips points are displayed on any plan view that the table is on.

The use a grip point, you must first click on the **Move** icon on the **Table Editor** toolbar and then click and accept on the **grip** that you wish to use.



## 18.4.4.2 Column and Row Edits



See

[18.4.4.2.1 Insert Row](#)

[18.4.4.2.2 Insert Column](#)

[18.4.4.2.3 Delete Row](#)

[18.4.4.2.4 Delete Column](#)

[18.4.4.2.5 Resize Row](#)

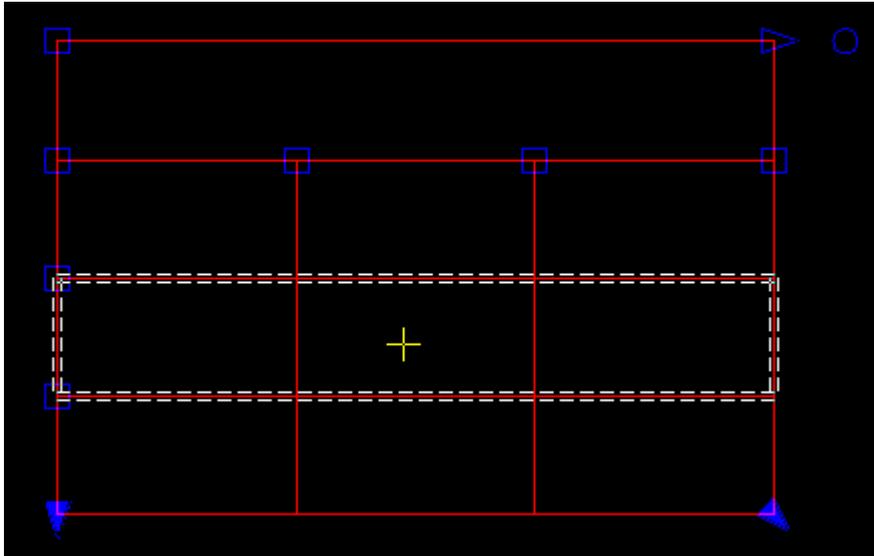
[18.4.4.2.6 Resize Column](#)

### 18.4.4.2.1 Insert Row

Click on the **Insert row** icon and the following message will appear in the screen message area.

<select row to insert (a)bove, (b)elow or a(p)pend> [picks][fast][Menu]

Then move over the table until the row you want to insert a row above or below highlights.



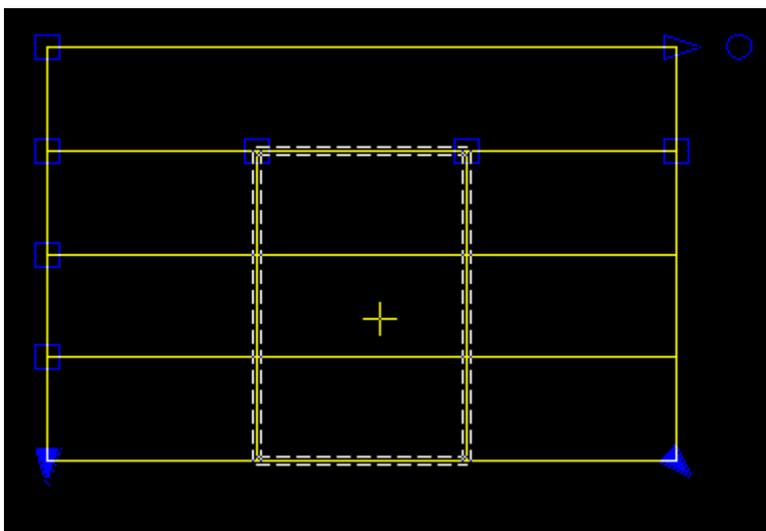
Press **a** or **A** to insert above, or **b** or **B** to insert below, the highlighted row.

### 18.4.4.2.2 Insert Column

Click on the **Insert column** icon and the following message will appear in the screen message area.

<select column to insert (l)eft, (r)ight or a(p)pend> [picks][fast][Menu]

Then move over the table until the column you want to insert a column either to the left or right of highlights.



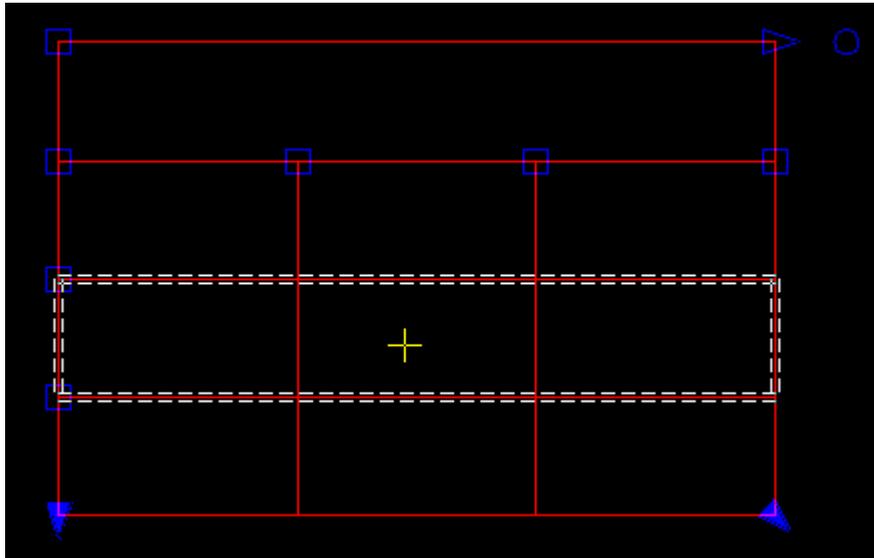
Press **l** or **L**, to insert to the left, or **r** or **R** to insert to the right, of the highlighted column.

### 18.4.4.2.3 Delete Row

Click on the **Delete row** icon and the following message will appear in the screen message area.

```
<select row to (d)elele> [picks][accepts][Menu] Cursor-position
```

Then move over the table until the row you want to delete highlights



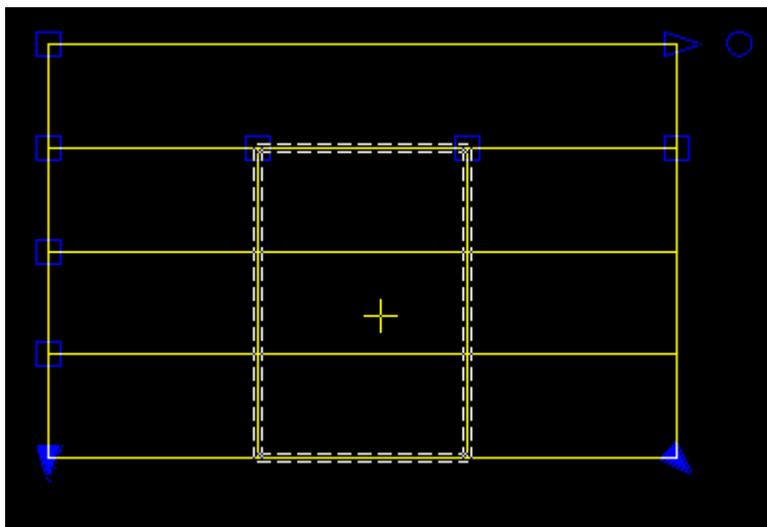
Press **d** or **D**, or pick and accept, to delete the highlighted row.

### 18.4.4.2.4 Delete Column

Click on the **Delete column** icon and the following message will appear in the screen message area.

```
<select column to (d)elele> [picks][accepts][Menu] Cursor-position
```

Then move over the table until the column you want to delete highlights



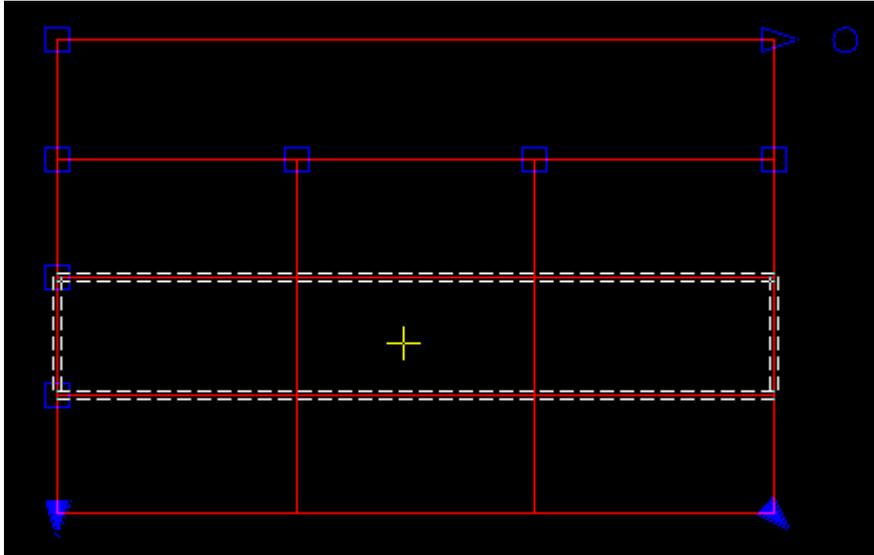
Press **d** or **D**, or pick and accept, to delete the highlighted column.

### 18.4.4.2.5 Resize Row

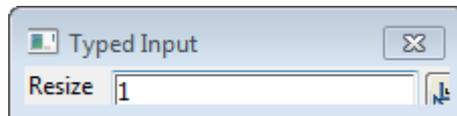
Click on the **Resize row** icon and the following message will appear in the screen message area.

```
<select row to re(s)ize> [picks][fast][Menu]
```

Then move over the table until the row you want to resize highlights.



Press **s** or **S**, or pick and accept, and the **Resize Typed Input** box will appear with the current size of the row in it.



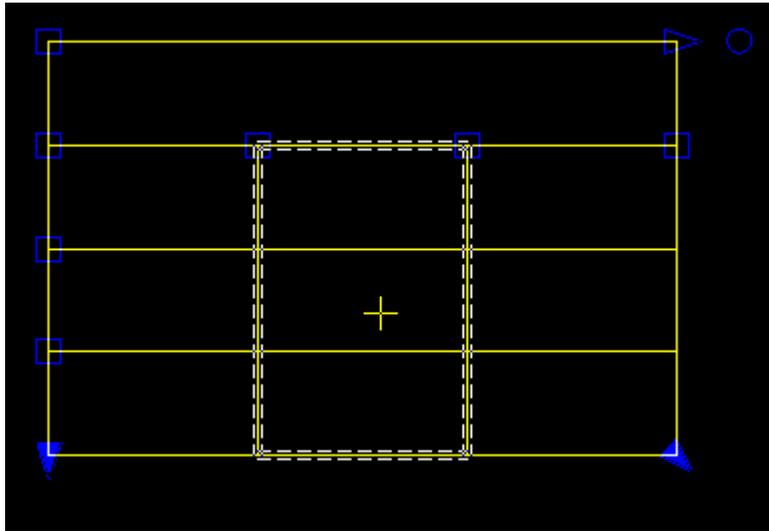
Type the new size into the box and then press <Enter>. The box will disappear and the row given the new size value.

### 18.4.4.2.6 Resize Column

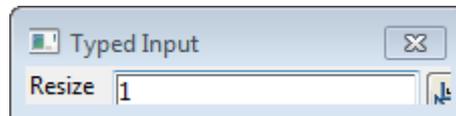
Click on the **Resize column** icon and the following message will appear in the screen message area.

```
<select column to re(s)ize> [picks][fast][Menu]
```

Then move over the table until the column you want to resize highlights.

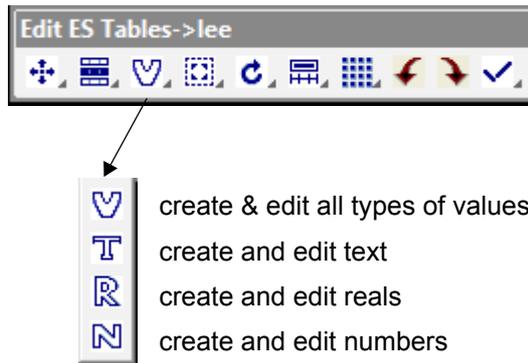


Press **s** or **S**, or pick and accept, and the *Resize Typed Input* box will appear with the current size of the selected column in it.



Type the new size into the box and then press <Enter>. The box will disappear and the column given the new size value.

### 18.4.4.3 Text and Values Edit



See

[18.4.4.3.1 Value Create and Edit](#)

[18.4.4.3.2 Text Create and Edit](#)

[18.4.4.3.3 Real Create and Edit](#)

[18.4.4.3.3 Real Create and Edit](#)

### 18.4.4.3.1 Value Create and Edit

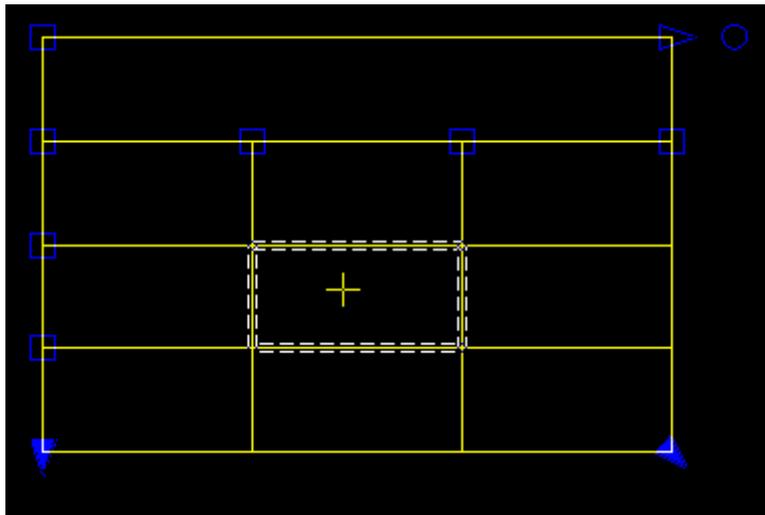
For an empty cell, **Value** can create either a text, number or real value.

If used on a non empty cell, the cell contents are displayed in a Typed input box or type Text, Number or Real depending on the type of the cell contents.

Click on the **Value** icon and the following message appears in the screen message area.

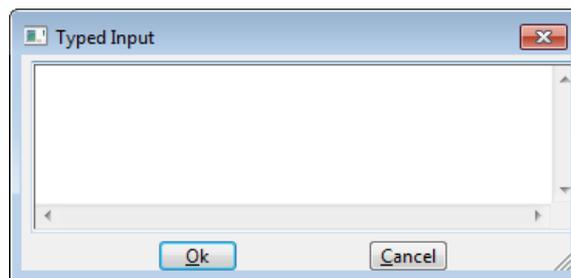
<select cell to edit (t)ext (n)umber (r)real> [picks][fast][Menu]

Move over the table until the cell you want to edit or create a value, highlights.

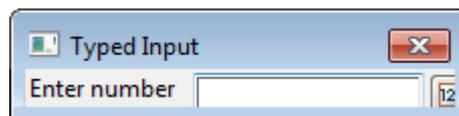


If the cell is **empty**,

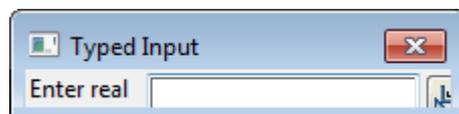
Type **t** or **T**, or pick and accept inside the cell, and the *Enter text* **Typed Input** box will appear.



Type **n** or **N**, and the *Enter number* **Typed Input** box will appear.



Type **r** or **R**, and the *Enter real* **Typed Input** box will appear.

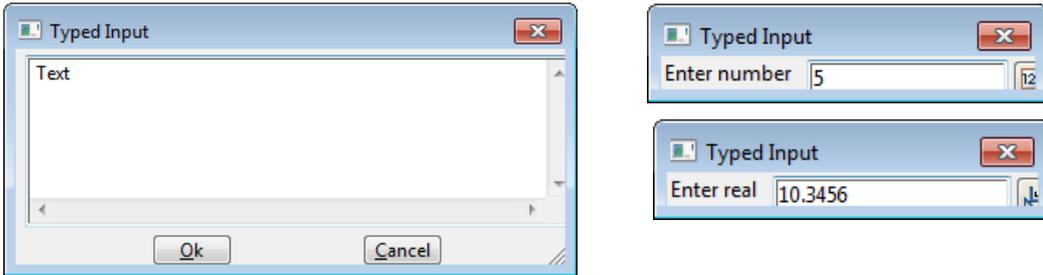


Type the new text, number or real into the box and press <Enter>.

The box will disappear and the new text, number or real placed in the cell, and the cell given the appropriate type.

Note: If the type is **Real** then the actual typed in value is kept in the cell BUT it is displayed with the number of decimal places given in the Table style.

If the cell is **not** empty, then the cell value is displayed in a **Typed Input** box of the **same type** as the **cell contents**.



Type the new **Text**, **Number** or **Real** into the **Typed Input** box and press <Enter>.

The box will disappear and the new value placed in the cell.

Note: If the type is **Real** then the actual typed in value is kept in the cell BUT it is displayed with the number of decimal places given in the Table style.

### 18.4.4.3.2 Text Create and Edit

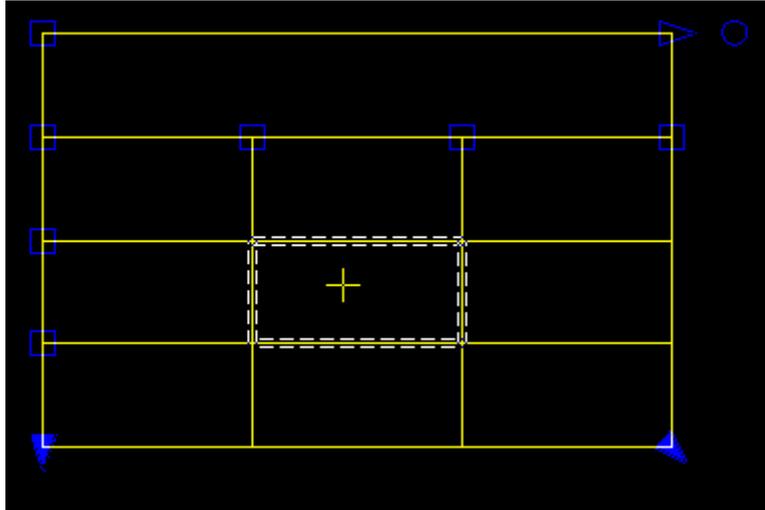
For an empty cell, **Text** creates and displays one or more lines of text.

If used on a non empty cell, the cell contents are displayed in a Text box, and when saved, is written into the cell as a **Text** type

Click on the **Text** icon and the following message will appear in the screen message area.

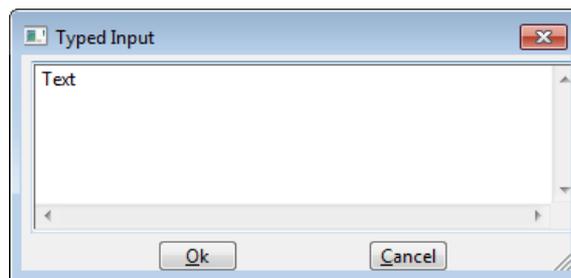
```
<select cell to edit (t)ext> [picks][fast][Menu]
```

Move over the table until the cell you want to edit or create a Text, highlights.



Type **t** or **T**, or pick and accept inside the cell, and the *Enter text* **Typed Input** box will appear.

If the cell was empty then nothing is in the *Enter text* box but if the cell is not empty, the contents of the cell are displayed as text.



Type the new text into the *Enter text* box and press <Enter>.

The box will disappear and the new text placed in the cell, and the cell given type **Text**.

### 18.4.4.3 Real Create and Edit

For an empty cell, **Real** creates and displays a real with the number of decimal places used in the display is taken from the Table style.

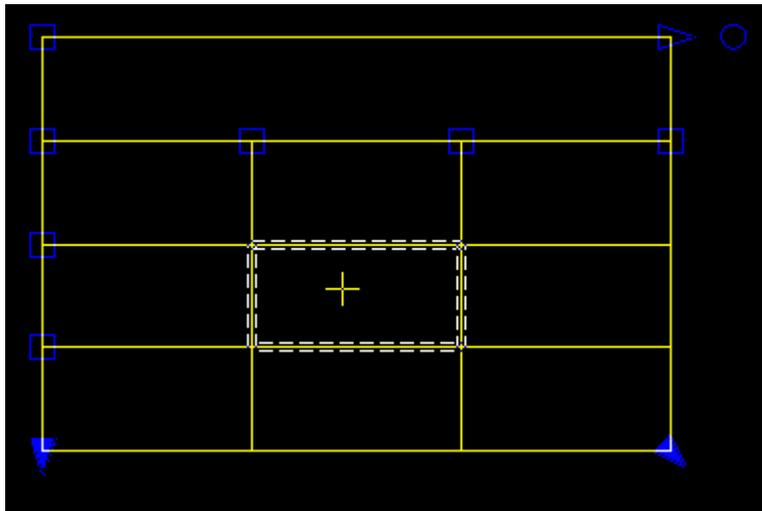
If used on a non empty cell, the cell contents are converted into a real and displayed in a Real box for editing, and when saved, is written into the cell as a **Real** type.

If the cell contains text that can't be converted to a real then the Real box is left empty.

Click on the **Real** icon and the following message will appear in the screen message area.

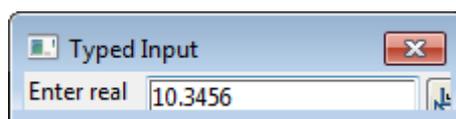
<select cell to edit (r)real> [picks][accepts][Menu] Cur

Move over the table until the cell you want to edit or create a Number, highlights.



Type **r** or **R**, or pick and accept inside the cell, and the *Enter real* **Typed Input** box will appear.

If the cell was empty then nothing is in the *Enter real* box but if the cell is not empty, the contents of the cell are displayed a Real, or nothing if the contents can't convert to a real.



Type the new real into the *Enter real* box and press <Enter>.

The box will disappear and the new real placed in the cell and displayed with the number of decimal places given in the Table style. The cell given type **Real**.

Note: If the type is **Real** then the actual typed in value is kept in the cell BUT it is displayed with the number of decimal places given in the Table style.

### 18.4.4.3.4 Number Create and Edit

For an empty cell, **Number** creates and displays a number (a positive or negative integer).

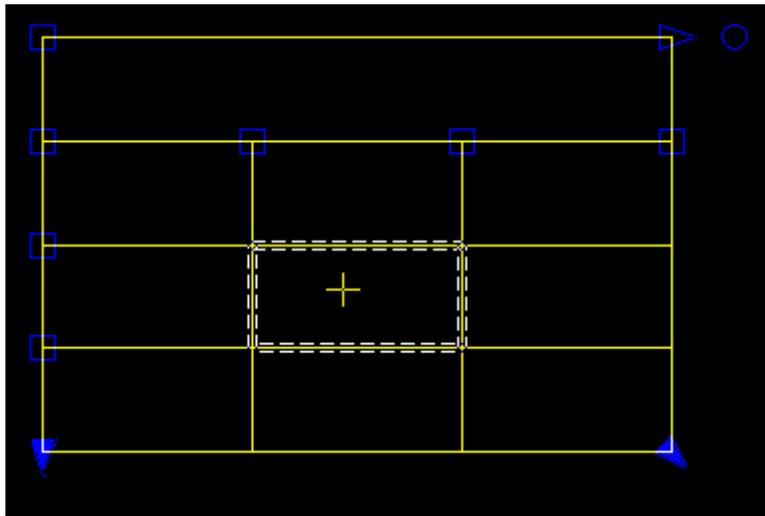
If used on a non empty cell, the cell contents are converted into a number and displayed in a number box for editing, and when saved, is written into the cell as a **Number** type.

If the cell contains text that can't be converted to a number then the Number box is left empty.

Click on the **Number** icon and the following message will appear in the screen message area.

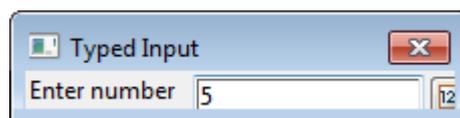
```
<select cell to edit (n)umber> [picks][fast][Menu]
```

Move over the table until the cell you want to edit or create a Number, highlights.



Type **n** or **N**, or pick and accept inside the cell, and the *Enter number* **Typed Input** box will appear.

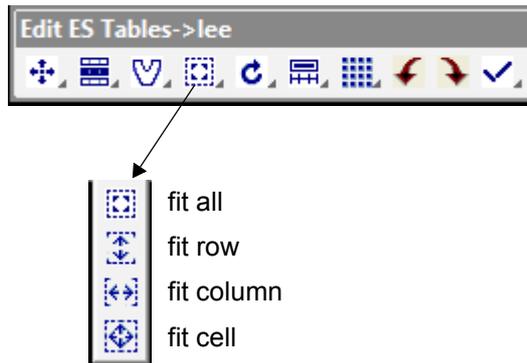
If the cell was empty then nothing is in the *Enter number* box but if the cell is not empty, the contents of the cell are displayed a Number, or nothing if the contents can't convert to a number.



Type the new number into the *Enter number* box and press <Enter>.

The box will disappear and the new number placed in the cell, and the cell given type **Number**.

## 18.4.4.4 Fit



See

[18.4.4.4.1 Fit All](#)

[18.4.4.4.2 Fit Row](#)

[18.4.4.4.3 Fit Column](#)

[18.4.4.4.4 Fit Cell](#)

### 18.4.4.4.1 Fit All

**Fit** goes through each cell and resizes it both vertically and horizontally so that the text fits inside the cell.

It then looks at each row and makes all the cells in the same row as high as the highest one in the row.

It then looks at each column and makes all the cells in the same column as wide as the widest one in the column.

**Fit** runs even if **Auto sizing** is set to **no**.

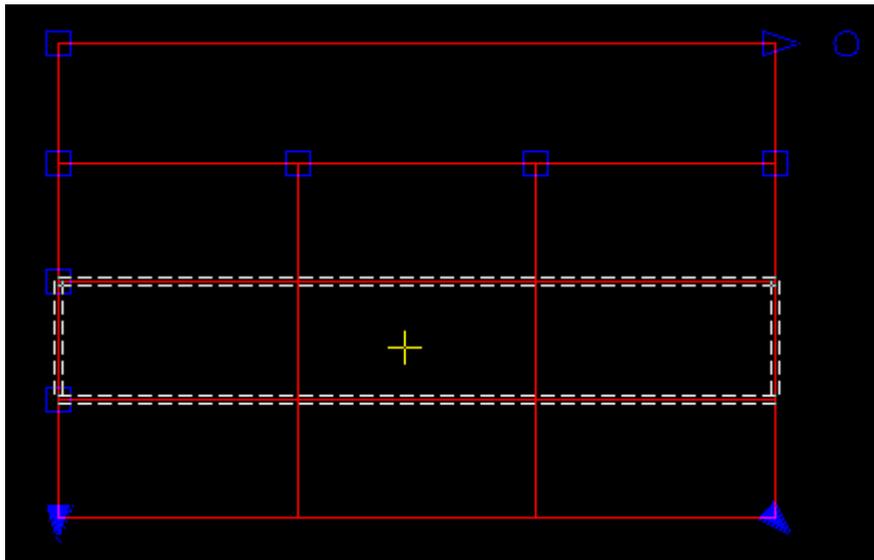
### 18.4.4.2 Fit Row

**Fit row** goes through each cell in the selected row to find the maximum height required to fit the text vertically in the cell, and then makes the height of the row the maximum of all the heights required for each cell. That is, it makes the row high enough to fit the text vertically in every cell in the row.

Click on the **Fit row** icon and the following message will appear in the screen message area.

```
<select row to fit> [picks][fast][Menu]
```

Then move over the table until the row you want to fit highlights.



Accepting makes the row high enough to fit any text in the row.

**Fit row** runs even if **Auto sizing** is set to **no**.

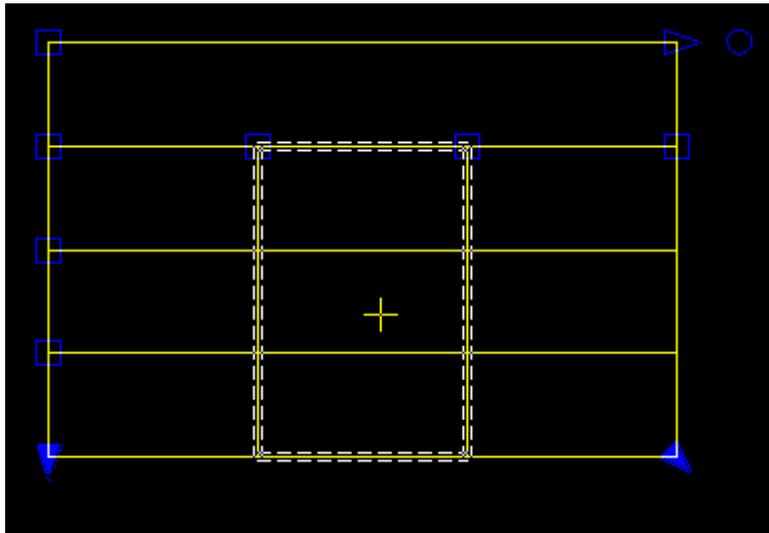
### 18.4.4.3 Fit Column

**Fit column** goes through each cell in the selected column to find the maximum width required to fit the text horizontally in the cell, and then makes the width of the column the maximum of all the widths required for each cell. That is, it makes the column wide enough to fit the text horizontally in every cell in the column.

Click on the **Fit column** icon and the following message will appear in the screen message area.

```
<select column to fit> [picks][fast][Menu]
```

Then move over the table until the column you want to fit highlights.



Accepting makes the column wide enough to fit any text in the column.

**Fit row** runs even if **Auto sizing** is set to **no**.

### 18.4.4.4 Fit Cell

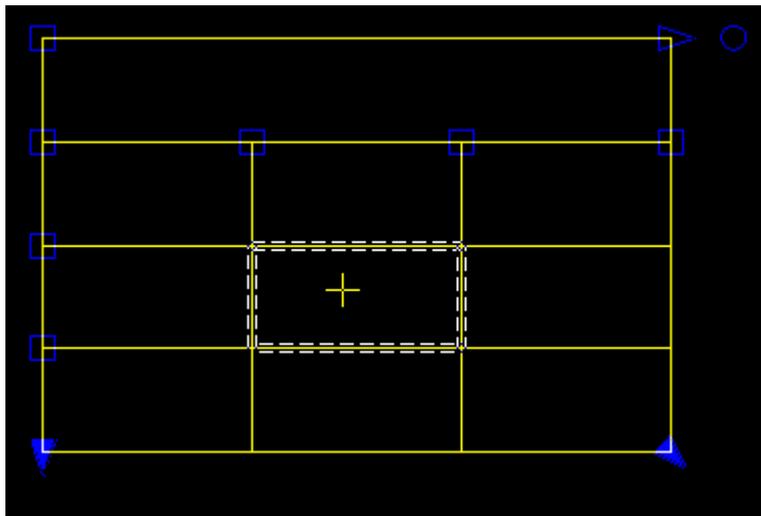
**Fit cell** increases the row height enough to fit the text vertically in the cell and also increases the column width enough to fit the text horizontally in the cell. That is, it makes the row and column wide enough to fit the text in the selected cell.

Note that the row height and column width will not be decreased, only increased

Click on the **Fit cell** icon and the following message will appear in the screen message area.

<select cell to fit> [picks][fast][Menu]

Then move over the table until the cell you want to fit highlights.



Accepting makes the column wide enough to fit any text in the column.

**Fit cell** runs if **Auto sizing** is set to **no**.

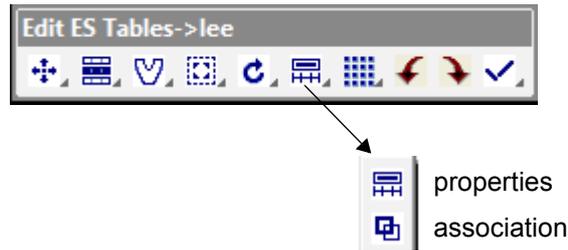
### 18.4.4.5 Recalc



Sometime when the object is **associated** with the **Table** is modified, the **Table** does not update.

**Recalc** forces an update of all the information connected with the **Association** object.

## 18.4.4.6 Properties and Associations



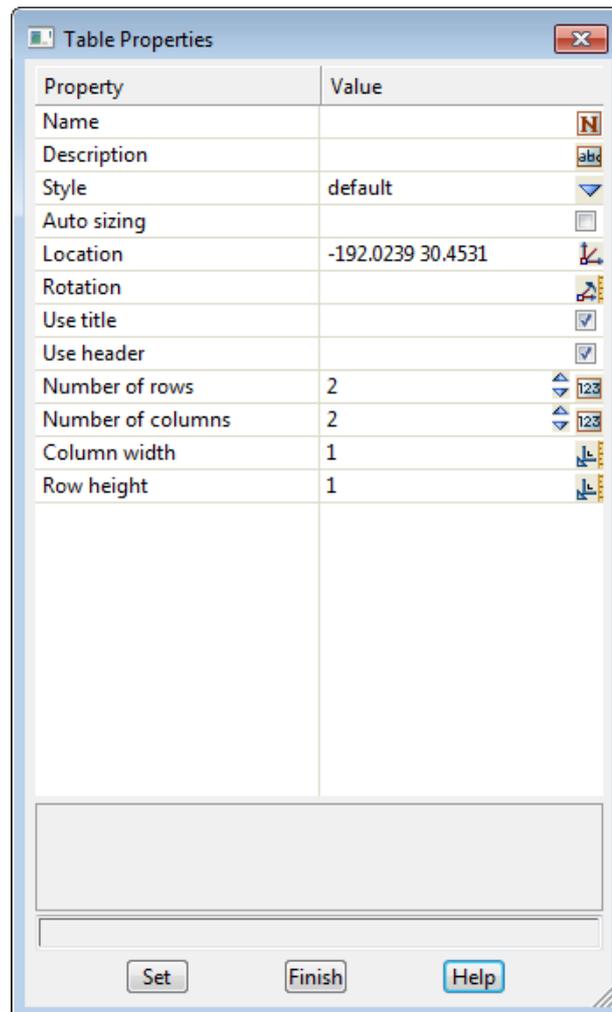
See

[18.4.4.6.1 Properties](#)

[18.4.4.6.2 Association](#)

### 18.4.4.6.1 Properties

Click on the **Properties** icon and the **Table Properties** panel comes up with the values for the current table in it.

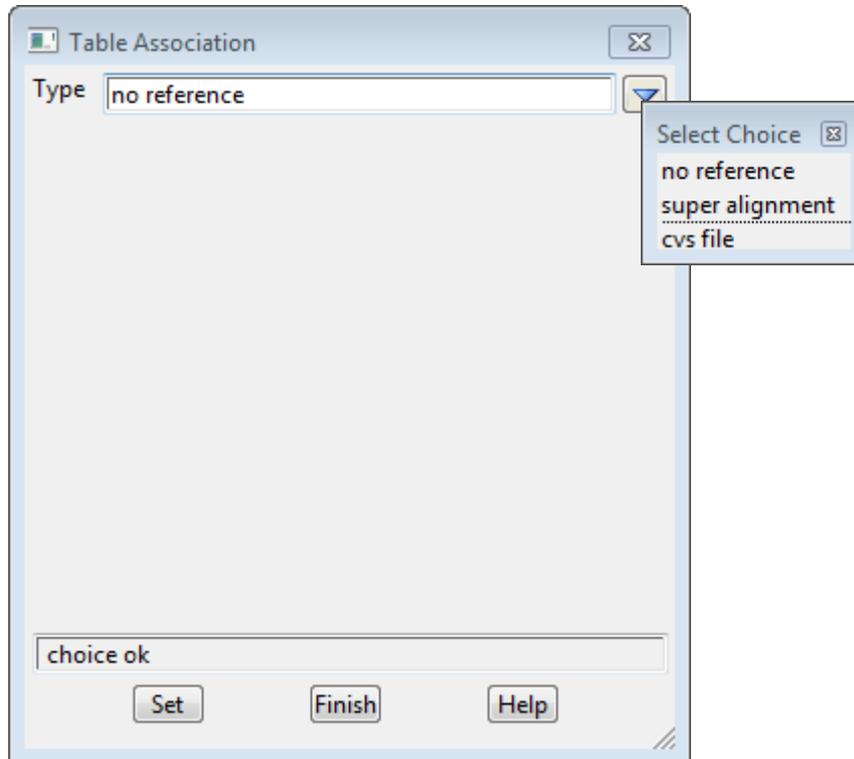


Change values in the panel and then click **Set** for them to become the tables new values.

### 18.4.4.6.2 Association

Click on the **Association** icon and the **Table Association** panel comes up.

If there is no Association for the table, one can be added with this option.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Type</b>	choice box	no reference, super alignment csv file	

*if **no reference**, then the table is not associated with anything.*

*If **super alignment**, read in and associate a CSV file with the table. See [18.4.4.6.2.1 Associate a Super Alignment](#)*

*If **CSV file**, read in and associate a CSV file with the table. See [18.4.4.6.2.2 Associate a CSV File](#)*

**Set** button

*set the association for the table.*

See

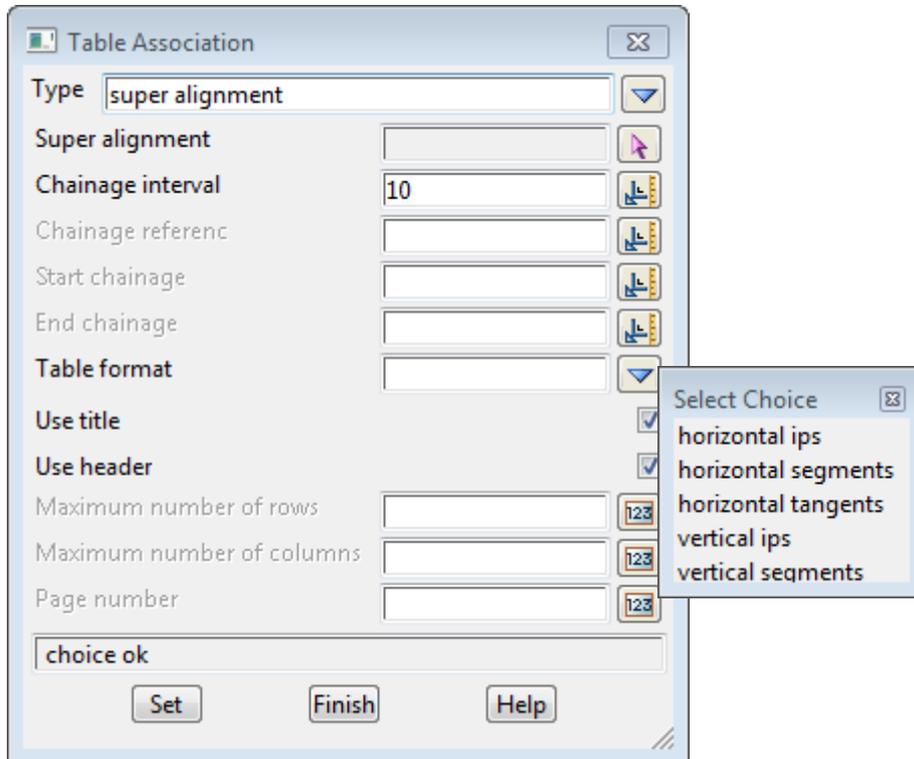
[18.4.4.6.2.1 Associate a Super Alignment](#)

[18.4.4.6.2.2 Associate a CSV File](#)

### 18.4.4.6.2.1 Associate a Super Alignment

Choice **super alignment** is selected when a super alignment is to be associated with the table.

When **super alignment** is selected, the panel changes to display fields to enter information about the super alignment and how it is to be associated with the table.

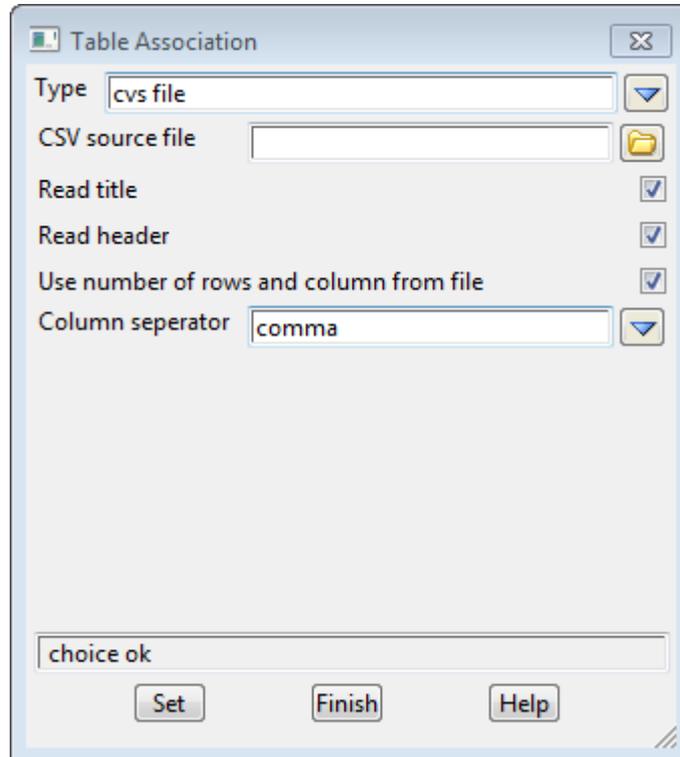


For information about the fields in this panel, see [18.4.2 Create Table from a Super Alignment](#).

#### 18.4.4.6.2 Associate a CSV File

Choice **CSV file** is selected when a CSV file is to be associated with the table.

When **CSV file** is selected, the panel changes to display fields to enter for information about the CSV file and how it is to be associated with the table.



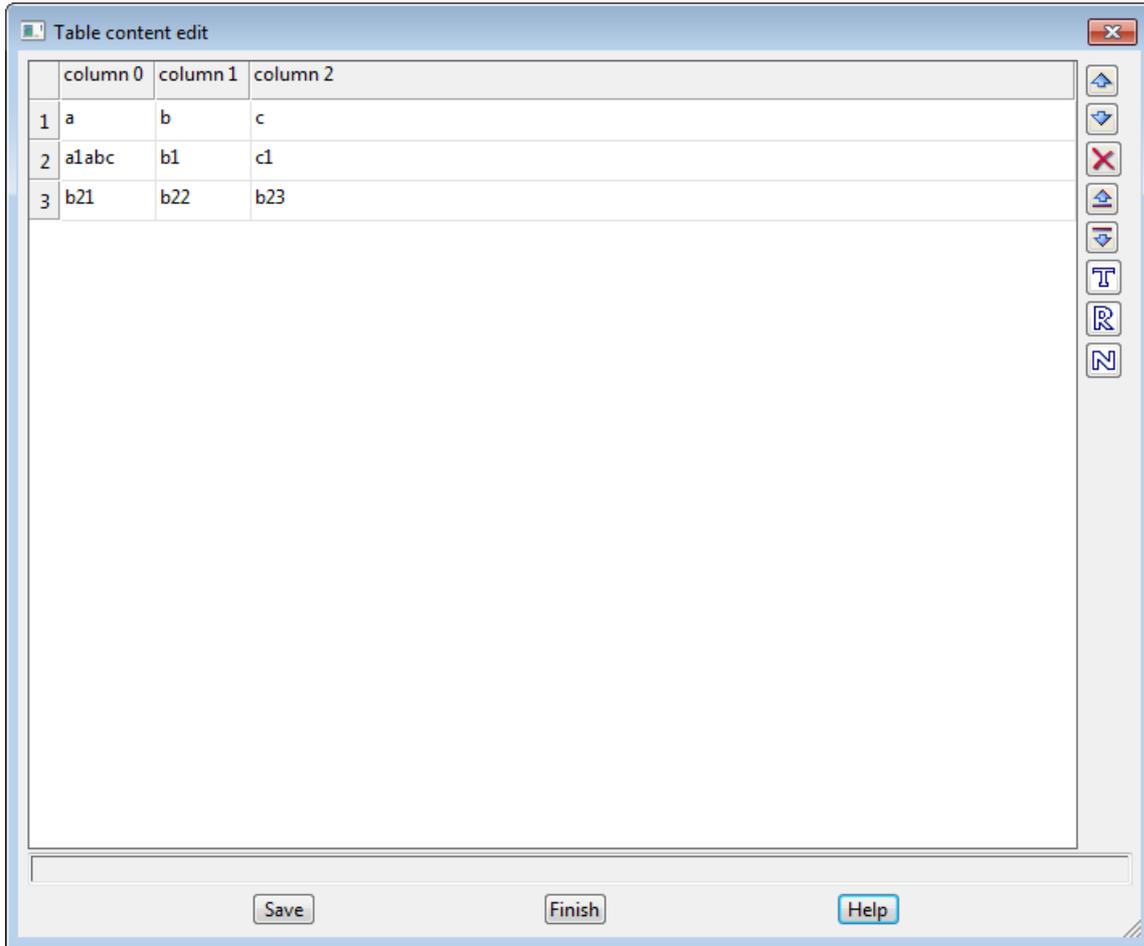
For information about the fields in this panel, see [18.4.3 Create Table from a CSV File](#).

### 18.4.4.7 Content



**content**

Click on the **Content** icon and the **Table Content Edit** panel comes up with the values for the current table in it.

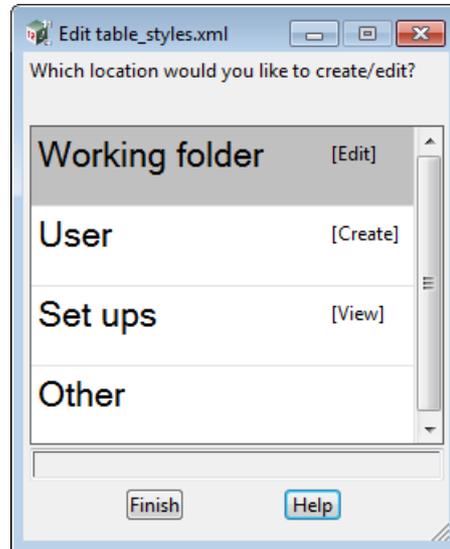


In the panel you can edit the values in each cell, add new values, add or delete rows or move rows up or down.

Finally click **Save** for the contents of the panel to become the new values in the table.

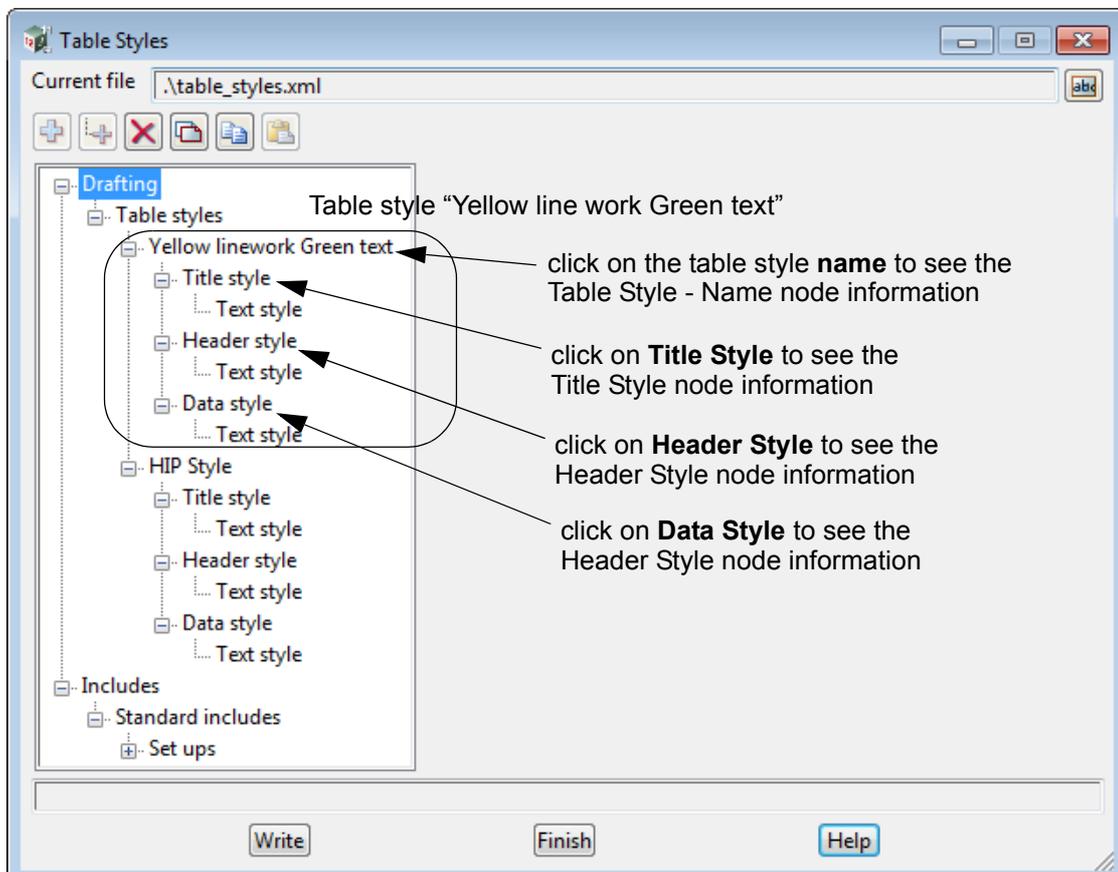
## 18.4.5 Table Styles

For information about where how to find and create table\_styles.xml  
 When you click on the option an **Edit table\_styles.xml** panel is brought up and the panel shows the standard areas for looking for the table\_styles.XML files - Working folder, Customer (User), User, Set Ups and Other.



For information about where how to find and create table\_styles.xml files, see [18.5.1 Create & Editing Dimensions, Leader and Table Styles](#).

Once a table\_styles.xml has been opened, the **Table Styles** pane is displayed.



See

[18.4.5.1 Table Styles - General Tab](#)

[18.4.5.2 Table Styles - Title Style Node](#)

[18.4.5.3 Table Styles - Header Style Node](#)

[18.4.5.4 Table Styles - Data Style Node](#)

### 18.4.5.1 Table Styles - General Tab

This is the screen you get when clicking on the **Table style** name.

Name	d1	
Description	test table style	

**Name** text box

*name of the table style.*

**Description** text box

*description of the table style.*

### 18.4.5.2 Table Styles - Title Style Node

This is the screen you get when clicking on the **Table style >Title style** node.

Cell linestyle	1	
Cell line colour	yellow	
Cell line weight	0	
Fill colour	no colour	
Fill blend	1	

**Cell linestyle** linestyle box available linestyles

*linestyle for drawing the line work around the title row in the table.*

**Cell line colour** text box

*colour of the line work drawn around the title row.*

**Cell line weight** real box

*weight of the line work drawn around the title row.*

**Fill colour** colour box available colours

*colour to fill the title row.*

**Fill blend** real box

*blend for the fill colour for the title row.*

To control the text in the Title row, see [18.4.5.3.1 Table Styles - Header Style > Text Style Node](#)

### 18.4.5.2.1 Table Styles - Title Style > Text Style Node

This is the screen you get when clicking on the **Table style >Title style >Text style** node.

If **Use title** is **yes** in the tables **Properties** then this node controls the drawing of the text in the Title at the top of the table.

Textstyle	Arial	T
Name		abc
Type	world	▼
Size	0.5	⏏
Angle	0°00'	⏏
X-factor	1.0	⏏
Slant	0°00'	⏏
Offset	0.0	⏏
Raise	0.0	⏏
Colour	green	■
Whiteout		■
Border		■
Border type		▼
Justify	middle-centre	▼
TTF Underline	no	▼
TTF Strikeout	no	▼
TTF Italic	no	▼
TTF Outline	no	▼
TTF Weight	400.0	⏏

These are the standard settings for defining text

### 18.4.5.3 Table Styles - Header Style Node

This is the screen you get when clicking on the **Table style >Header style** node.

Cell linestyle	1	
Cell line colour	yellow	
Cell line weight	0	
Fill colour	no colour	
Fill blend	1	

**Cell linestyle**                      linestyle box                                      available linestyles

*linestyle for drawing the line work around the header row in the table.*

**Cell line colour**                      text box

*colour of the line work drawn around the header row.*

**Cell line weight**                      real box

*weight of the line work drawn around the header row.*

**Fill colour**                                      colour box                                      available colours

*colour to fill the header row cells.*

**Fill blend**                                      real box

*blend for the fill colour for the header row cells.*

To control the text in the Header row, see [18.4.5.3.1 Table Styles - Header Style > Text Style Node](#)

#### 18.4.5.3.1 Table Styles - Header Style > Text Style Node

This is the screen you get when clicking on the **Table style >Title style >Text style** node.

If **Use header** is **yes** in the tables **Properties** then this node controls the drawing of the text in the **Header** at the top of each column in the table.

Textstyle	Arial	
Name		
Type	world	
Size	0.5	
Angle	0°00'	
X-factor	1.0	
Slant	0°00'	
Offset	0.0	
Raise	0.0	
Colour	green	
Whiteout		
Border		
Border type		
Justify	middle-centre	
TTF Underline	no	
TTF Strikeout	no	
TTF Italic	no	
TTF Outline	no	
TTF Weight	400.0	

These are the standard settings for defining text

### 18.4.5.4 Table Styles - Data Style Node

This is the screen you get when clicking on the **Table style >Data style** node.

Cell linestyle	1	
Cell line colour	yellow	
Cell line weight	0	
Fill colour	no colour	
Fill blend	1	

**Cell linestyle**                      linestyle box                                      available linestyles

*linestyle for drawing the line work around the data rows in the table.*

**Cell line colour**                      text box

*colour of the line work drawn around the data rows.*

**Cell line weight**                      real box

*weight of the line work drawn around the data rows.*

**Fill colour**                                      colour box                                      available colours

*colour to fill the data row cells.*

**Fill blend**                                      real box

*blend for the fill colour for the data row cells.*

To control the text in the Data rows, see [18.4.5.3.1 Table Styles - Header Style > Text Style Node](#)

#### 18.4.5.4.1 Table Styles - Data Style > Text Style Node

This is the screen you get when clicking on the **Table style >Data style Text style** node.

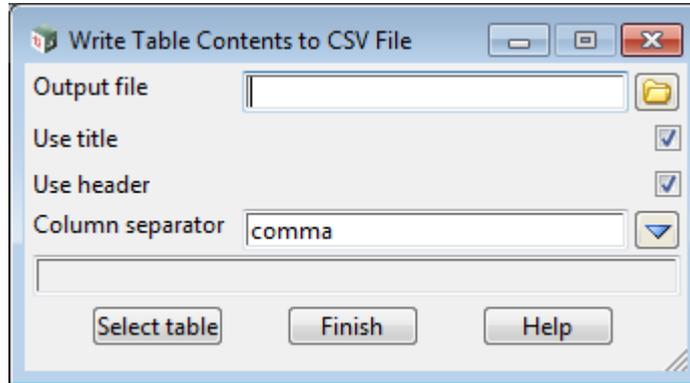
This node controls the drawing of the text in the **Data** rows in the table.

Textstyle	Arial	
Name		
Type	world	
Size	0.5	
Angle	0°00'	
X-factor	1.0	
Slant	0°00'	
Offset	0.0	
Raise	0.0	
Colour	green	
Whiteout		
Border		
Border type		
Justify	middle-centre	
TTF Underline	no	
TTF Strikeout	no	
TTF Italic	no	
TTF Outline	no	
TTF Weight	400.0	

These are the standard settings for defining text.

## 18.4.6 Table to CSV

This option creates a CSV file for a table that is associated with a CSV file. Selecting the **Table to csv** brings up the **Write Table Contents to CSV File** panel.



The fields and buttons used in this panel have the following functions.

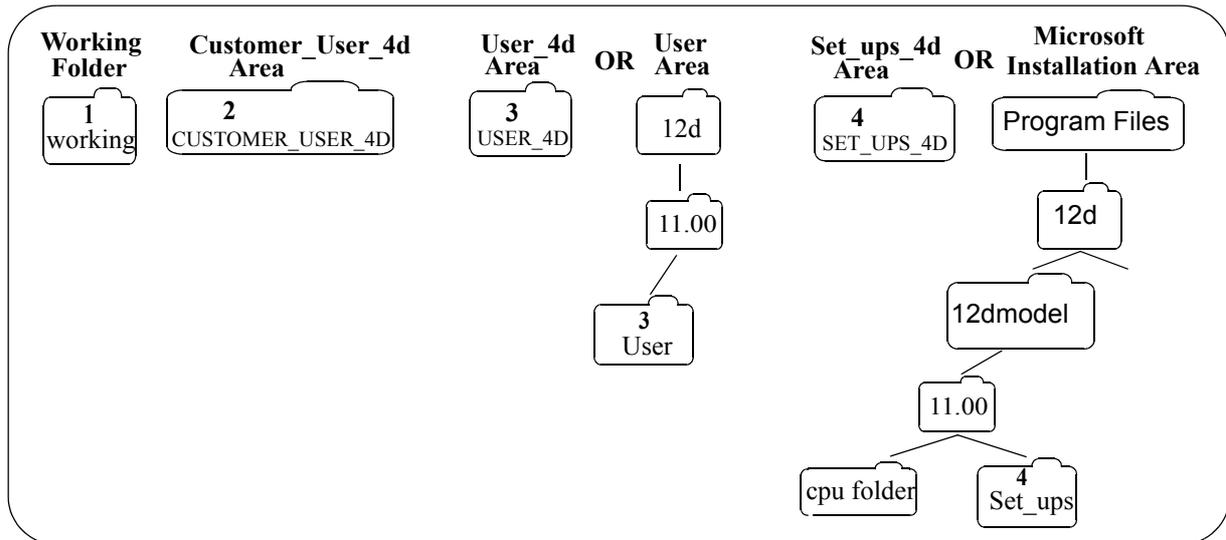
Field Description	Type	Defaults	Pop-Up
<b>Output file</b> <i>name of the CSV file to write to the table out to.</i>	file box		*.csv files
<b>Use title</b> <i>if <b>ticked</b>, the first line of the CSV file is the title of the table.</i>	tick box		
<b>Use header</b> <i>if <b>ticked</b>, the next line of the CSV file is the Headers for each column of the table.</i>	tick box		
<b>Column separator</b> <i>the column separator used when writing out the CSV file.</i>	choice box	commas	comma, semicolon, tab
<b>Select table</b> <i>select the table AND then write out the contents of the table to a CSV file.</i>	button		

## 18.5 Style XML Files for Dimensions, Leaders and Tables

There are XML files to define the styles used for Dimensions, Leaders and Tables:

- (a) for **Dimensions**, the file is **dimensions\_styles.xml**
- (b) for **Leaders**, the file is **leader\_styles.xml**.
- (c) for **Tables**, the file is **table\_styles.xml**.

These XML files are all **Set Up** files like *colours.4d* and folders are searched in a specific order to find the files.



**HOWEVER**, unlike *colours.4d*, it doesn't matter if the appropriate styles xml file is found in one folder, the search **continues** and **all** the **folders** that **contain the file** are found.

And, again unlike *colours.4d*, the styles used are the **merger** of **all** the **found files**.

The search order is still important because if the **same name** exists in more than one file in the found folders, the **first** occurrence in the files in the search order is the one that **is used**.

For example, if you had a dimensions style called **d1** in the file *dimensions\_styles.xml* in your working folder, and one called **d1** in the file *dimensions\_styles.xml* in Programs files\12d\12dmodel\11.0\set\_ups, then the **d1** in your **working folder** is the one that will be used and seen in any dimensions styles pop ups in **12d Model**.

See

[18.5.1 Create & Editing Dimensions, Leader and Table Styles](#)

## 18.5.1 Create & Editing Dimensions, Leader and Table Styles

**Position of option on menu:** CAD =>Dimensions =>Styles

**Position of option on menu:** CAD =>Leader =>Styles

**Position of option on menu:** CAD =>Table =>Styles

When you click on the options, a **Edit Styles** panel is brought up for the appropriate style and it shows the standard areas for looking for the Styles XML files - Working folder, Customer (User), User, Set Ups and Other - and displays whether the appropriate xml file:

(a) exists and available for editing (**Edit**)

If the xml file exists in the folder and you have access to edit it, then the folder name is written in the panel and **[Edit]** is written on the right hand side of the line.

(b) does not exist but can be created (**Create**) by the editor

If the xml file does not exist in the folder and you have access to create it, then the folder name is written in the panel and **[Create]** is written on the right hand side of the line.

In this case a new file is created and written out to the given folder.

(c) is in a folder that has no access for editing and so can only be viewed (**View**)

If there is no access the folder to create/edit the file, then the folder name is written in the panel and **[View]** is written on the right hand side of the line. Although the styles can't be edited, they can be **copied**.

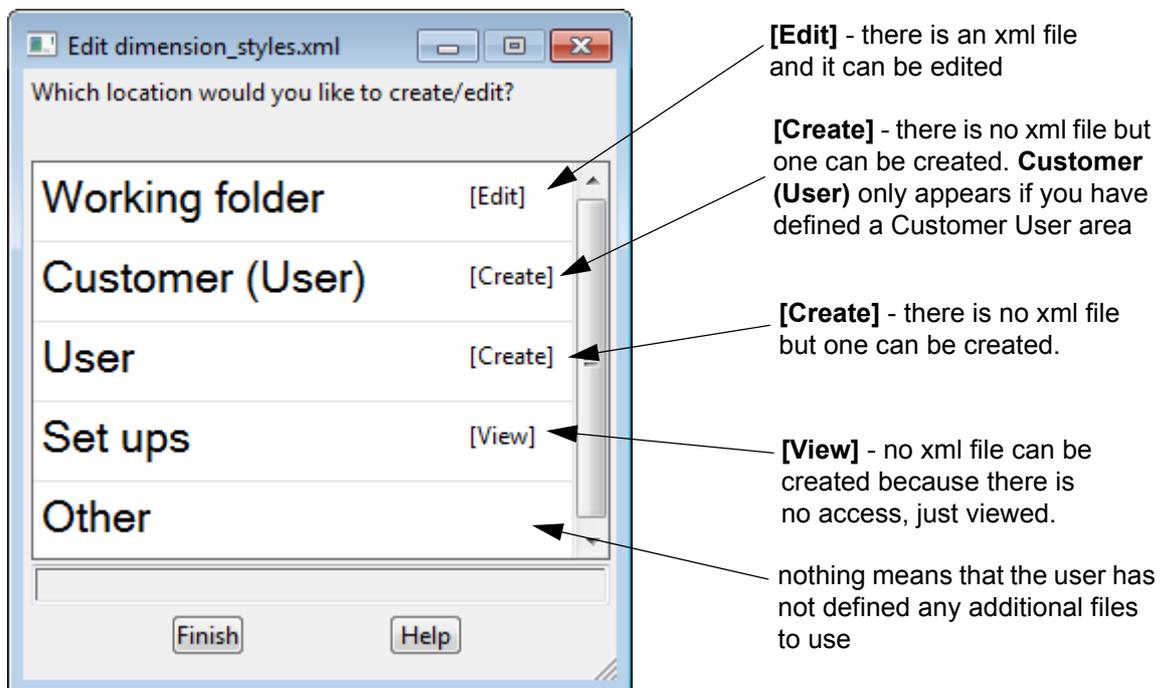
This usually applies to the **Set Ups** folder which normally needs Admin privileges to write to.

(d) is in **Customer (User)**.

This only appears if a *Customer User* area has been defined.

(e) is defined separately by the user and then **Other** is displayed.

In this case, the full path name of the XML file is given by the user.



See

[18.5.1.1 \[Create\] for Dimensions, Leader and Table Styles](#)

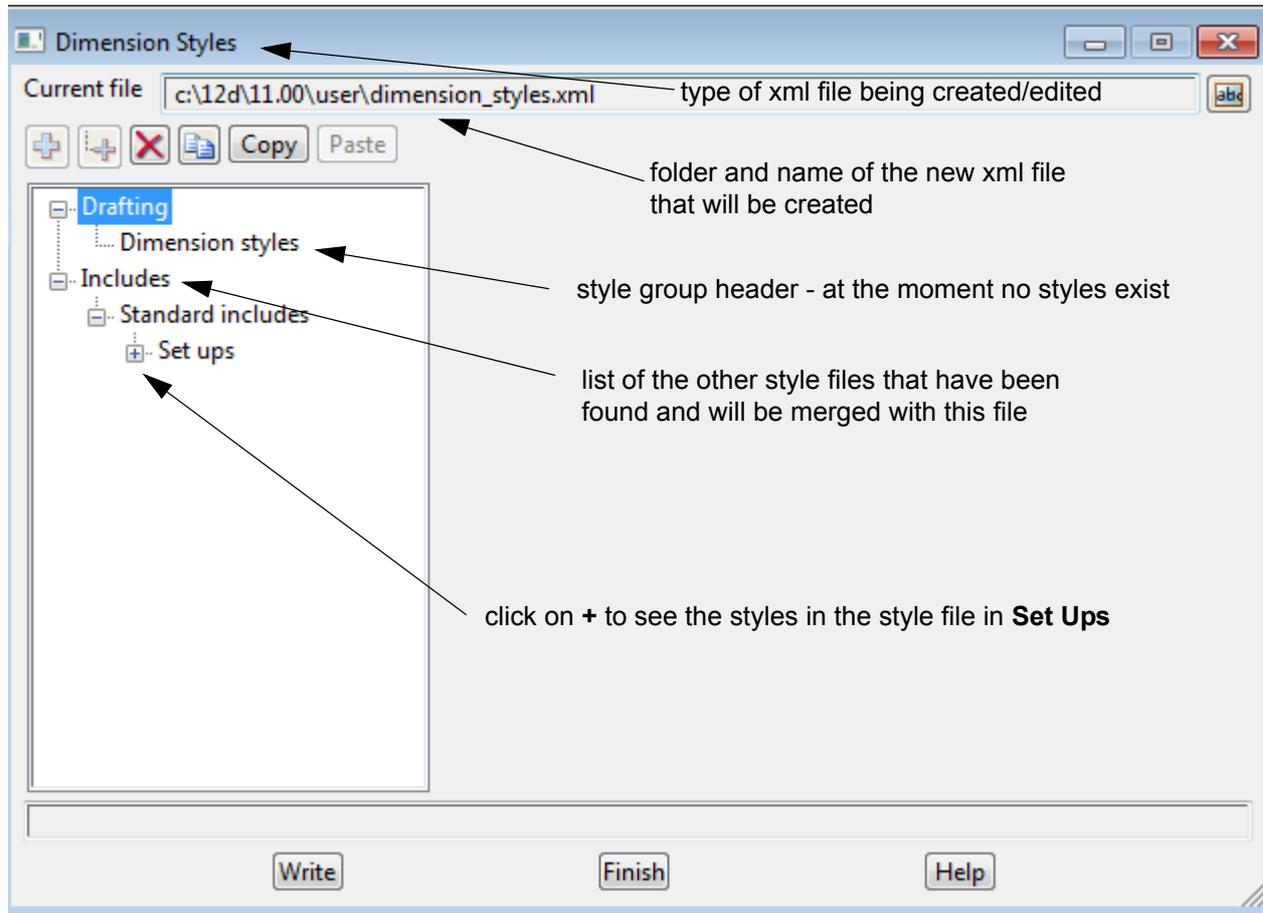
[18.5.1.2 \[Edit\] for Dimensions, Leader and Table Styles](#)

### 18.5.1.1 [Create] for Dimensions, Leader and Table Styles

When **[Create]** is shown, a new xml file can be created in that folder.

Click on the line with **[Create]** and the **Editor** for that Style starts up with a **style group header** but no styles in it.

All the other style files that are found are listed in the **Includes** section displayed at the bottom of the tree.



The included files can not be edited whilst create/editing this file BUT you can copy items from them and paste them into items in this file.

New styles can now be created (see [18.5.1.2.1 Creating New Styles](#)) and a new XML created by clicking on the **Write** button.

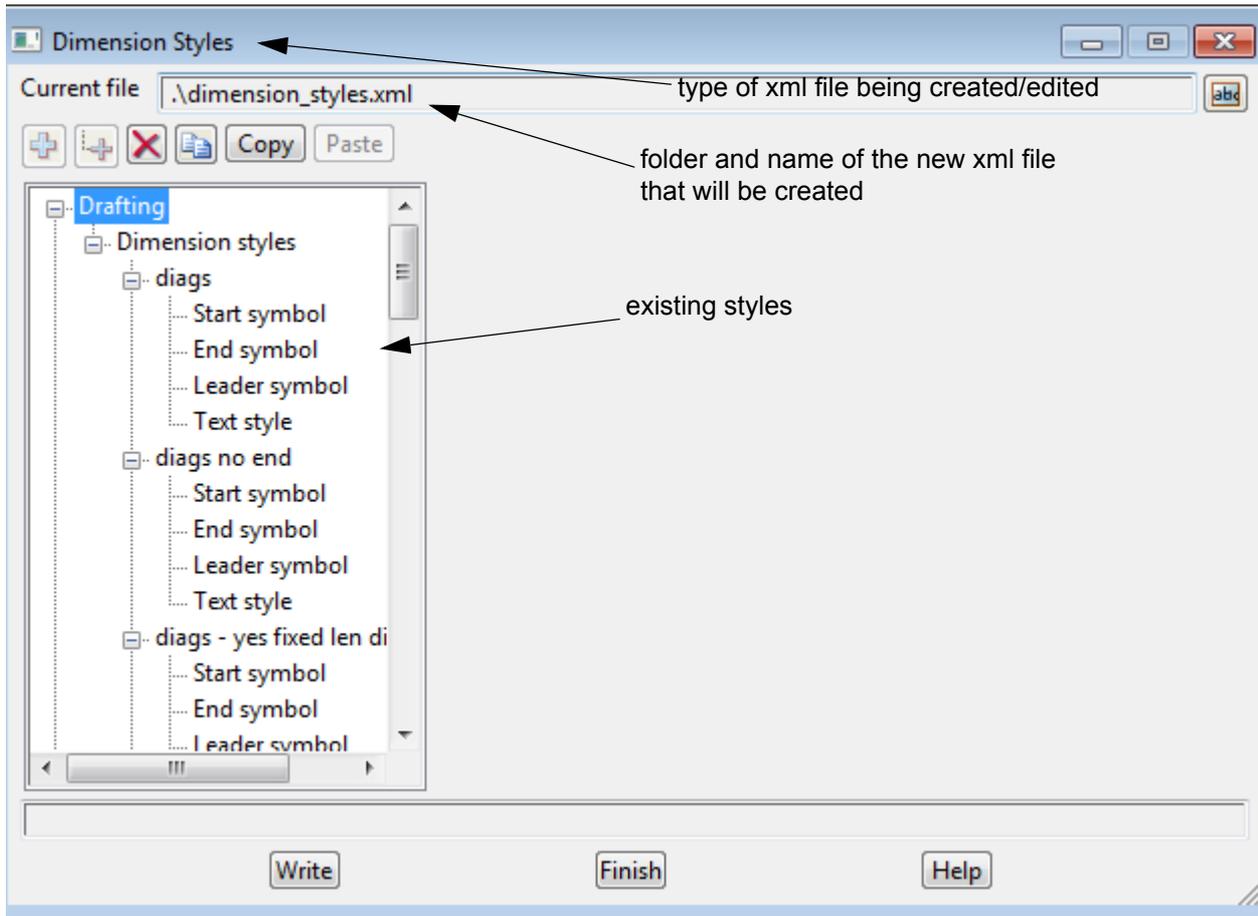
Creating and editing of styles is the same as for **[Edit]** and so is documented in that section. See [18.5.1.2 \[Edit\] for Dimensions, Leader and Table Styles](#).

**Write** - writes out the information in the panel to the files name given in **Current file**.

### 18.5.1.2 [Edit] for Dimensions, Leader and Table Styles

When **[Edit]** is shown, an xml file already exists and can be edited.

Click on the line with **[Edit]** and the **Editor** for that Style starts up with all the existing styles in that file displayed.



After completing any editing, the XML is saved by clicking on the **Write** button.

As in the **[Create]** case, all the other appropriate files that are found are listed in the Includes section are displayed at the bottom of the tree (see [18.5.1.1 \[Create\] for Dimensions, Leader and Table Styles](#)).

The included files can not be edited whilst create/editing this file BUT you can copy items from them and paste them into items in this file.

For editing in the panel, see

[18.5.1.2.1 Creating New Styles](#)

[18.5.1.2.2 Copy and Paste](#)

[18.5.1.2.3 Deleting a Style](#)

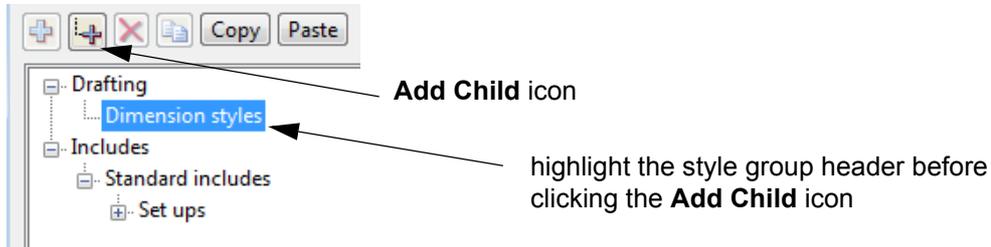
[18.5.1.2.4 Duplicating a Style](#)

[18.5.1.2.5 Editing a Style](#)

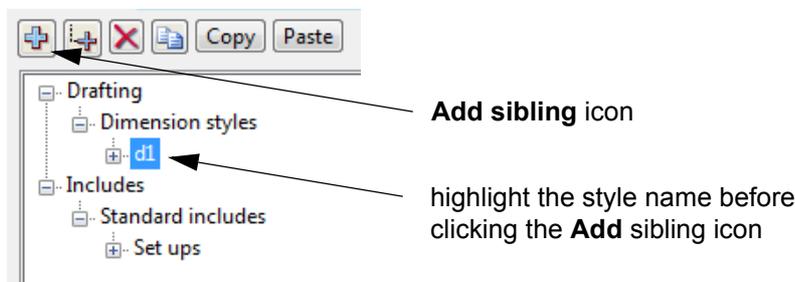
### 18.5.1.2.1 Creating New Styles

To create a **new style** from scratch when there are **no styles** already, highlight the **style group header** (Dimension styles, Leader styles or Table styles) and click on the **Add Child** icon.

**Note** - the **Add Child** icon adds a node as a subnode of the highlighted node.

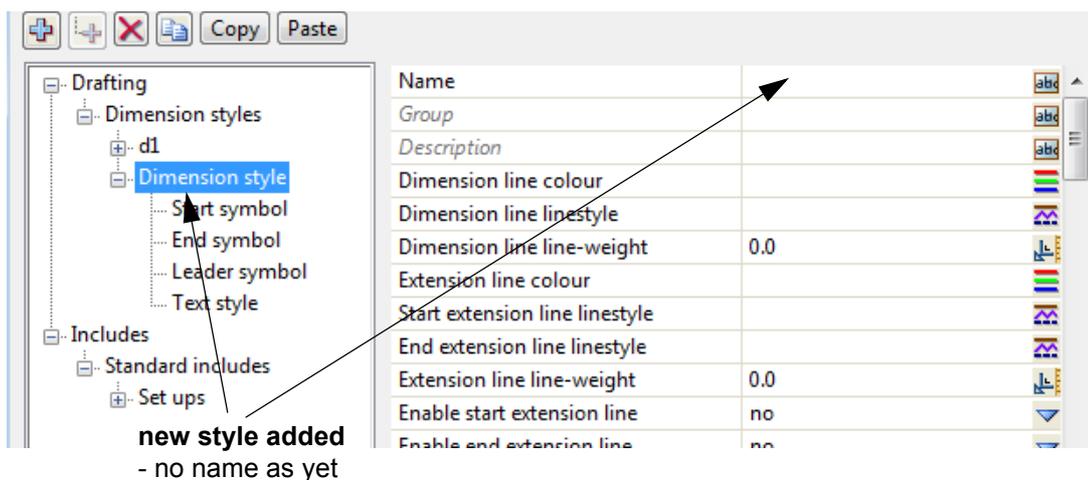


To create a **new style** from scratch when **some styles already exist**, you can highlight the **style name** (name of the style or Dimension style, Leader style or Table style if it has not been given a name) and click on the **Add** (Add Sibling) icon.

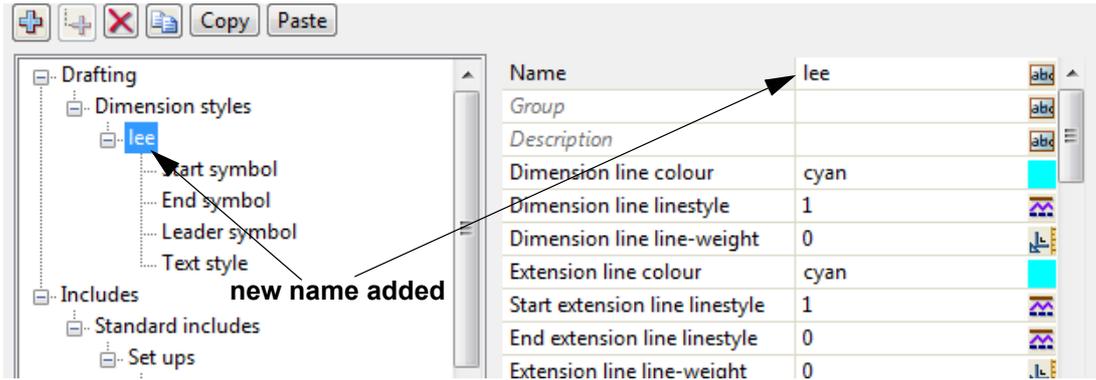


The **Add** icon adds a new item at **the same level** (a sibling) as the highlighted item and that is why a style must already exist so that one can be selected.

In both cases a new style is created with no name and so the general style name (Dimension style, Leader style or Table style) is written in the style list until the style is given a name.



Once the style is given a name then that name will appear in the tree.



Once the new style is created then it can be edited by going into each field of the style and making changes.

### Quicker Methods for Creating New Styles

When you click on a node to edit that section of the tree, you can not leave the edited node until all the required values are filled in. Similarly you won't be able to save the style unless all the required values are filled in. And this may involve filling in a lot of values.

So rather than starting from scratch, it is often easier to use **Duplicate** ([18.5.1.2.4 Duplicating a Style](#)) if there are already existing styles in the file, or **Copy** and **Paste** ([18.5.1.2.2 Copy and Paste](#)).

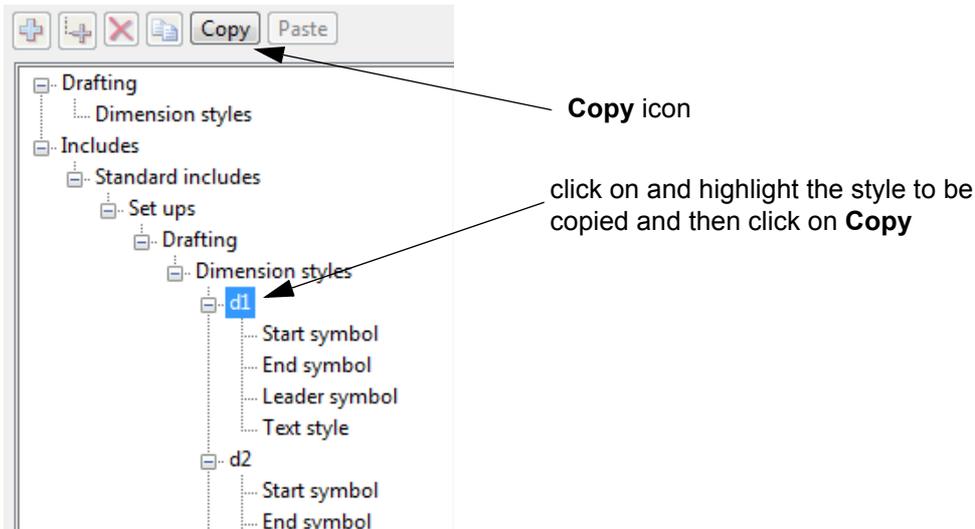
**Copy** and **Paste** is particularly useful when there are **no** styles in the file you are creating styles but there are styles in the other files listed in the **Includes** node. Although these styles can't be edited, they can be copied.

### 18.5.1.2.2 Copy and Paste

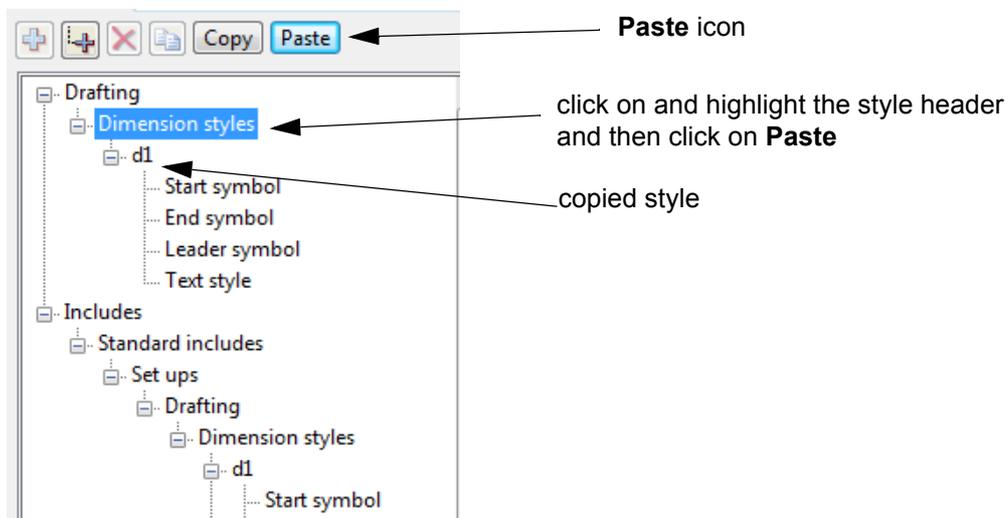
Rather than creating a new style from scratch, it is often easier to use **Copy** and **Paste** to create a copy of an existing style, and then edit the copied style.

Although there may be no styles in the folder you are creating styles for, there will usually be some styles in the XML file in **Set Ups** that can be copied.

In the tree in the **Editor**, click on the **+** in front of **Set Ups** to expand the tree and then click on the item in the tree that you want to copy. Then click on the **Copy** icon.



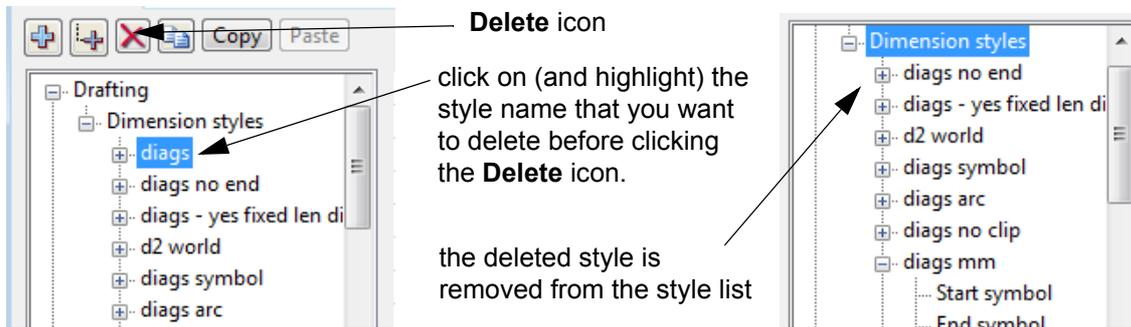
Next click on and highlight the style header (Dimension styles, Leader styles or Table styles) and click on **Paste**. The style will then appear under the style header with the same name as the original style. The copied style can then be edited.



**NOTE:** If the name of the copied style is **not changed** then this new style will be used instead of the one in **Set Ups**.

### 18.5.1.2.3 Deleting a Style

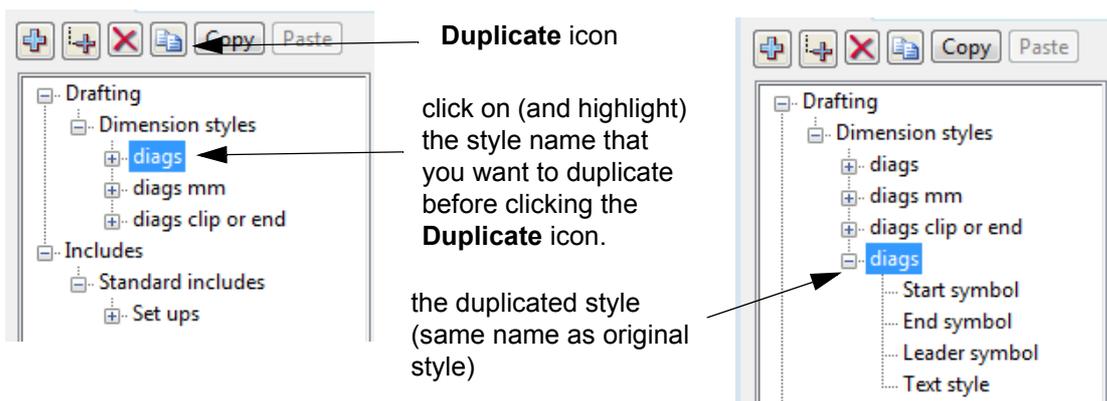
To **delete** a style, simply click on and highlight the **style name**, and then click on the **Delete** icon.



### 18.5.1.2.4 Duplicating a Style

To **duplicate** a style, simply click on and highlight the **style name**, and then click on the **Duplicate** icon.

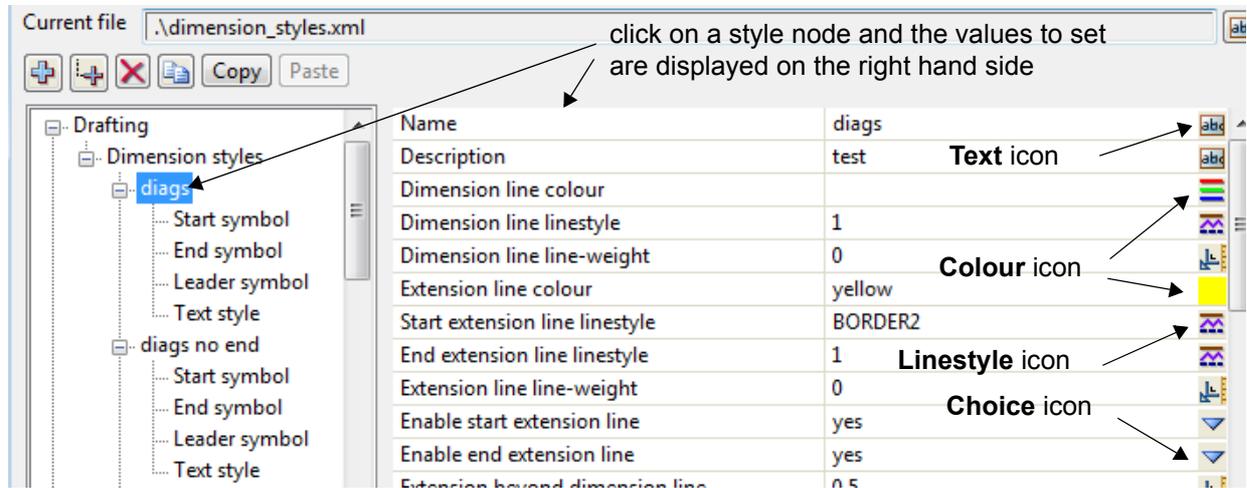
A copy of the style, with exactly the same name, is created and added to the end of the list of styles.



The name of the style needs to be changed and any other edits done that are required.

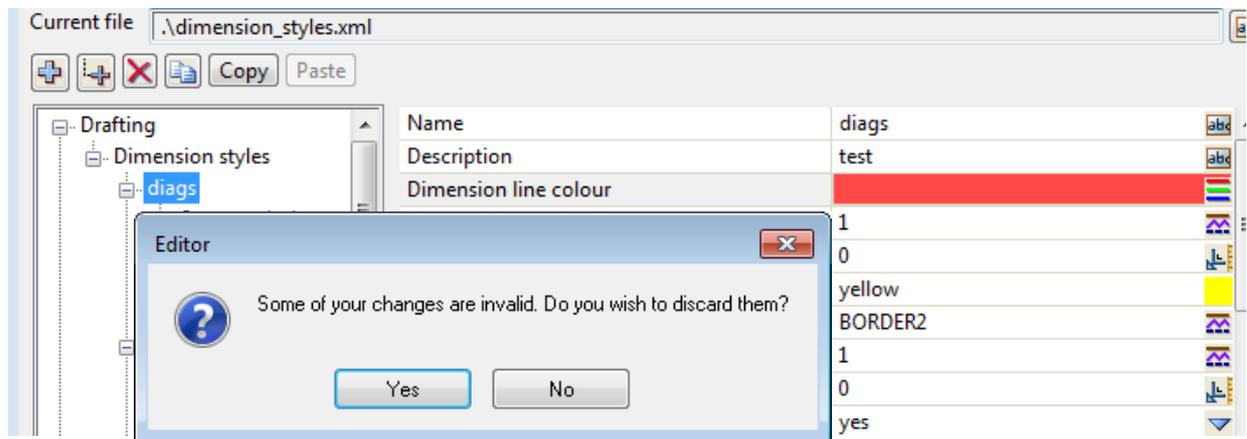
### 18.5.1.2.5 Editing a Style

To edit a **style**, you click on the style name, or expand the style and click on any of the sub nodes, and all the values for that node will be displayed on the right hand side of the panel.



The panel fields on the right hand side of the panel have the same icons as in other **12d Model** panels and the standard pop ups are all available. For example, text, colour and choice icons.

When clicking to go to another node, all the fields currently on the right hand side are validated and if there are any problems, the first line with an error is coloured red and an **Editor error** box is displayed.



If **Yes** is selected, then all the **invalid field** values are **discarded**, the **Editor error** panel removed, and the data for the new selected node displayed on the right hand side of the panel.

If **No** is selected, then the right hand side of the panel does not change and can undergo further edits.

# 19. Tins Menu

See

[19.1 127 Tins in a Super Tin](#)

[19.2 Exact Calculations Flag for a Super Tin](#)

[19.3 Colour by Triangle Data](#)

[19.4 New Tin Manager](#)

[19.5 New Recalc Super Tin](#)

[19.6 New Depth, Contour and Label Function](#)

[22. Design Menu](#)

[22.6.2 Shadow Analysis](#)

## 19.1 127 Tins in a Super Tin

The number of tins in a super tin has been increased from 62 to 127.

## 19.2 Exact Calculations Flag for a Super Tin

For a Super Tin, there is now an **Exact calculations flag** and if this is set to **on**, extra calculations are done when creating a Super Tin (and every time a tin included in a Super Tin is modified) so that the Super Tin can be used for Exact Volumes and Visualisation.

So Super Tins with the **Exact calculations** flag set support sections, Apply MTFs, site lines, and volumes by end area methods, as well as volumes by exact methods, and can also be used in visualisations.

**Note** - the reason that **Exact calculations** may be turned off is that for some tins, it may add significant processing time whenever a tin in the Super Tin is changed. So if that is slowing down your project and you don't need exact volumes or visualisations all the time, turn off **Exact calculations** for the Super Tin. **Exact calculations** can always be turned back on when you need them.

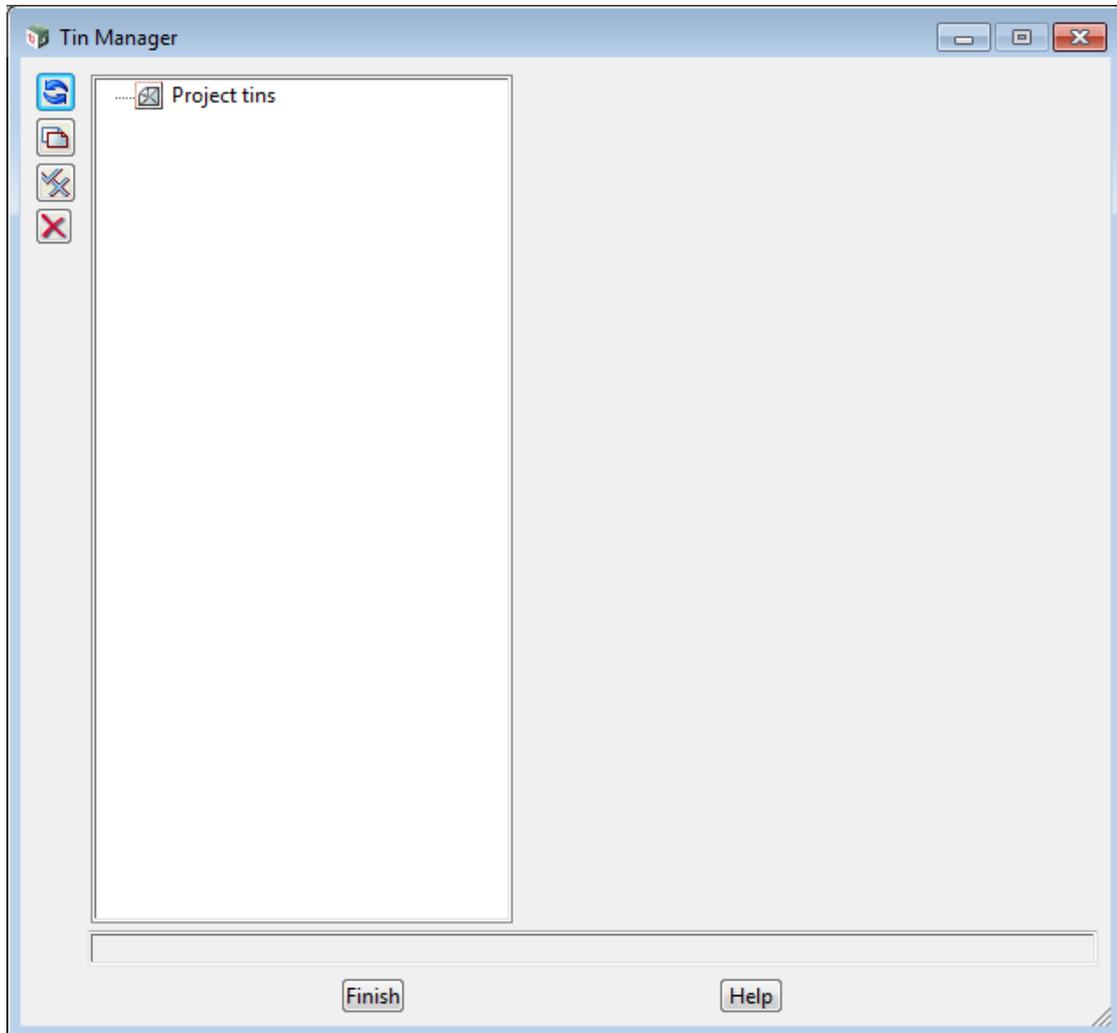
## 19.3 Colour by Triangle Data

When creating a Tins and using the tin is being created from data that is already triangles (the Triangle data method), there is now a **Colour by triangle data** tick box and it is ticked on, the triangles in the new tin are given the same colour at the each of the triangles in data being triangulated.

## 19.4 New Tin Manager

**Position of option on menu:** Tins =>Tin Manager

Selecting Tins=> Tin Manager displays the **Tin Manager** panel.



## 19.5 New Recalc Super Tin

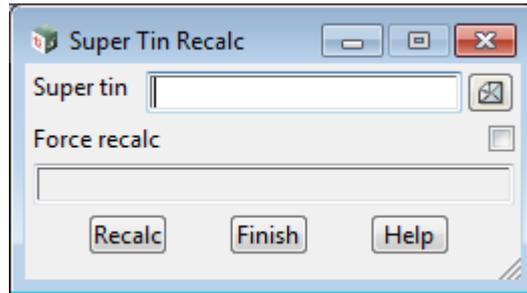
**Position of option on menu:** Tins =>Edit =>Recalc supertin

This option gets a super tin that has **Exact calculations** ticked on, to recalculate itself.

That is, the super tin updates all the interfaces with its component tins if there were any changes to the component tins.

The option can be used in a Chain.

Selecting Edit=> Recalc supertin displays the **Super Tin Recalc** panel.



The fields and buttons used in this panel have the following functions.

Field	Description	Type	Defaults	Pop-Up
<b>Super tin</b>	<i>name of the Super Tin to be recalculated.</i>	super tin box		existing Super Tins
<b>Force recalc</b>		tick box		
	<i>if ticked and <b>Exact calculations</b> is ticked on for the super tin, a recalc of the super tin is done.</i>			
	<i>If not ticked and <b>Exact calculations</b> is ticked on for the super tin, a recalc of the super tin is only done if the super tin believes it is necessary.</i>			
	<i>If <b>Exact calculations</b> is not ticked on for the super tin, then no recalc occurs.</i>			
<b>Recalc</b>		button		
	<i>recalculates the super tin</i>			

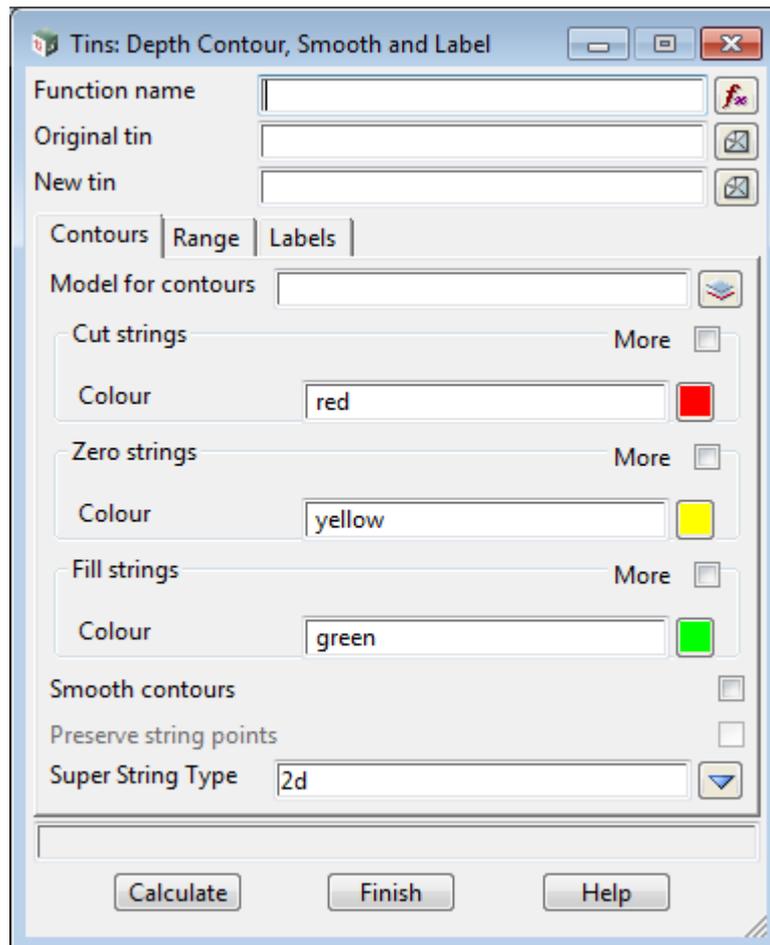
# 19.6 New Depth, Contour and Label Function

There is a new function to create depth contours and also label and smooth them.

**Tins =>Contour =>Depth Contour, Smooth and Label**

**Position of option on menu:** Tins =>Contour =>Depth contour, smooth and label

Selecting **Depth contour, smooth and label** displays the **Tins: Depth Contour, Smooth and Label** panel.



The fields and buttons used in this panel have the following functions.

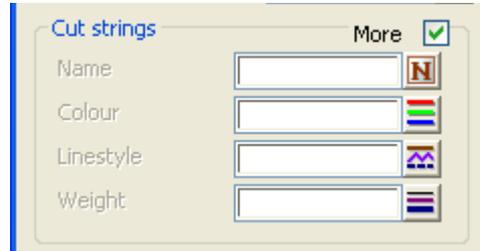
Field Description	Type	Defaults	Pop-Up
<b>Function name</b> <i>name of the depth contour, smooth and label function.</i>	function box		available depth contour fns
<b>Original/New tin</b> <i>name of the original/new tin to contour between. Cut is when the new tin is below the original tin.</i>	tin box		available tins

## Contours tab

### Cut/Zero/Fill strings sections

<b>More</b> <i>if not tick then only the Colour panel field is displayed.</i> <i>If not ticked then the extra fields Name, Colour, Linestyle are Weight are shown.</i>	tick box	not ticked
--	----------	------------

**Colour**                      colour box                      cyan                      available colours  
*if non-blank, colour for the strings*  
*If More is ticked then Name, Colour, Linestyle and Weight fields are displayed*



**Name**                      name box                      available names  
*if not blank, name to give the cut/zero/fill strings.*

**Colour**                      colour box                      cyan                      available colours  
*if not-blank, colour for the cut/zero/fill strings*

**Linestyle**                      linestyle box                      1                      available linestyles  
*if not blank, linestyles for the cut/zero/fill strings.*

**Weight**                      weight box  
*if not blank, weight to give the cut/zero/fill strings.*

**Smooth contours**                      tick box  
*if ticked, the contours are smoothed.*

**Preserve string points**                      tick box  
*if ticked, then the smoothed string goes through the original vertices of the non smoothed contour strings.*  
*If not ticked, then the smoothed strings do not have to include the vertices from the non-smoothed contours.*

**Super string type**                      choice box                      2d                      2d, 3d original, 3d new  
*If 2d, the created strings are 2d super strings with a z-value equal to the depth.*  
*If 3d original, the depth strings are draped over the original tin to form 3d super strings*  
*If 3d new, the depth strings are draped over the new tin to form 3d super strings.*

**Range tab**

**Start level**                      real box                      -10  
*the minimum depth to start the depth contours at.*

**End level**                      real box                      10  
*the maximum depth to calculate the depth contours to.*

**Interval**                      real box                      1  
*the interval between the depth contours.*

**Colour by range**                      tick box  
*if ticked, then the contours are coloured according to a depth range file rather than cut and fill colours.*  
*See XX.*

**Depth range file**                      depth range file box                      \*.drf



# 20 Drainage, Sewer and Tuflow

The **What's New for Drainage, Sewer and Tuflow** has not yet been documented.  
There will be **Seminars** and **Webinars** on the What's New for these items.

# 21 Volumes

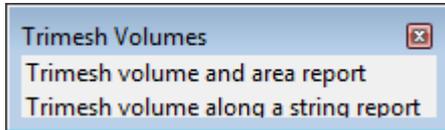
See

[21.1 Trimesh](#)

## 21.1 Trimesh

These volumes are calculated for trimeshes.

The **Trimesh** walk-right menu is



volumes of trimeshes

volumes of trimeshes for chainage interval along a string

For the options see:

[21.1.1 Trimesh Volume and Area Report](#)

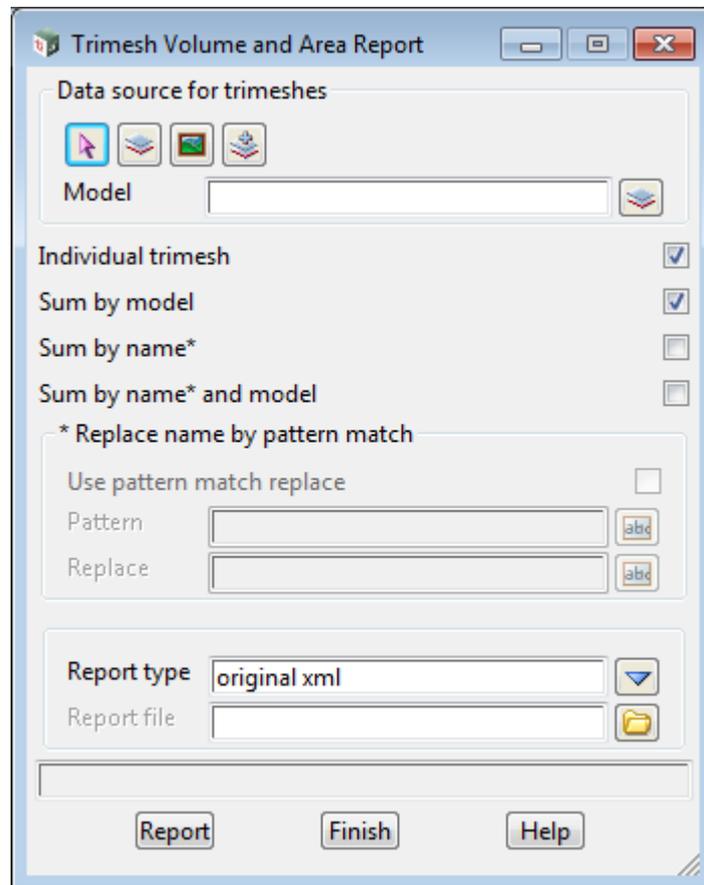
[21.1.2 Trimesh Volume Along a String](#)

## 21.1.1 Trimesh Volume and Area Report

**Position of option on menu:** Design =>Volumes =>Trimesh =>Trimesh volume report

This option calculate the volume of closed trimeshes, and user selected combinations of closed trimeshes.

On selecting **Trimesh volume report**, the **Trimesh Volume Report** panel is displayed.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>Data to search</i>			
<b>Data source</b>	input		
<i>source of trimeshes to calculate the volumes for.</i>			
<b>Individual trimesh</b>	tick box		
<i>if <b>ticked</b>, the volume for each individual trimesh will be written to the XML file.</i>			
<b>Sum by name</b>	tick box		
<i>if <b>ticked</b>, the sum of the volumes for trimeshes with the same name will be written to the XML file.</i>			
<b>Sum by model</b>	tick box		
<i>if <b>ticked</b>, the sum of the volumes for trimeshes in the same model will be written to the XML file.</i>			
<b>Sum by name and model</b>	tick box		
<i>if <b>ticked</b>, the sum of the volumes for trimeshes in the same model will be written to the XML file.</i>			

**Report type** choice box original xml, <Customize>  
*output format for the report.*

*For information on setting up custom reports from the generated XML file using xslts.*

**Report file** file box  
*an XML file will be created and a report of this name, and of the type given by **Report type** will be generated from the XML file. If the file already exists, a panel with **Replace** and **Cancel** buttons will be displayed.*

**Report** button  
*the volumes of the various combinations of trimeshes are calculated and written to the XML file.*

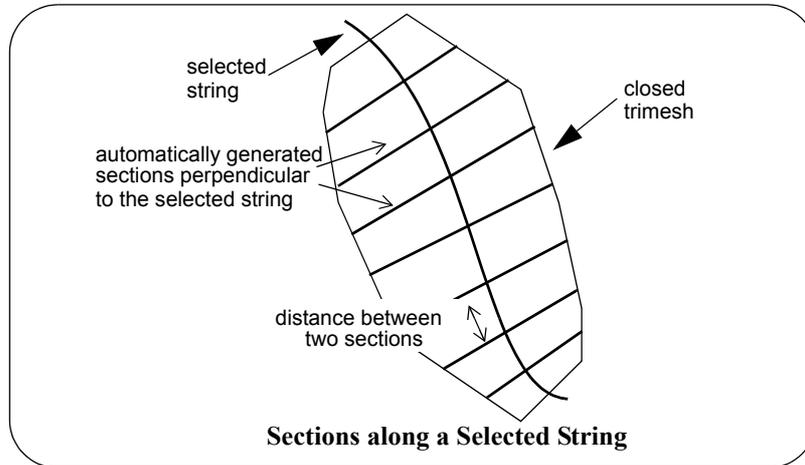
Continue to [21.1.2 Trimesh Volume Along a String](#) or return to [17.6 Trimesh](#).

## 21.1.2 Trimesh Volume Along a String

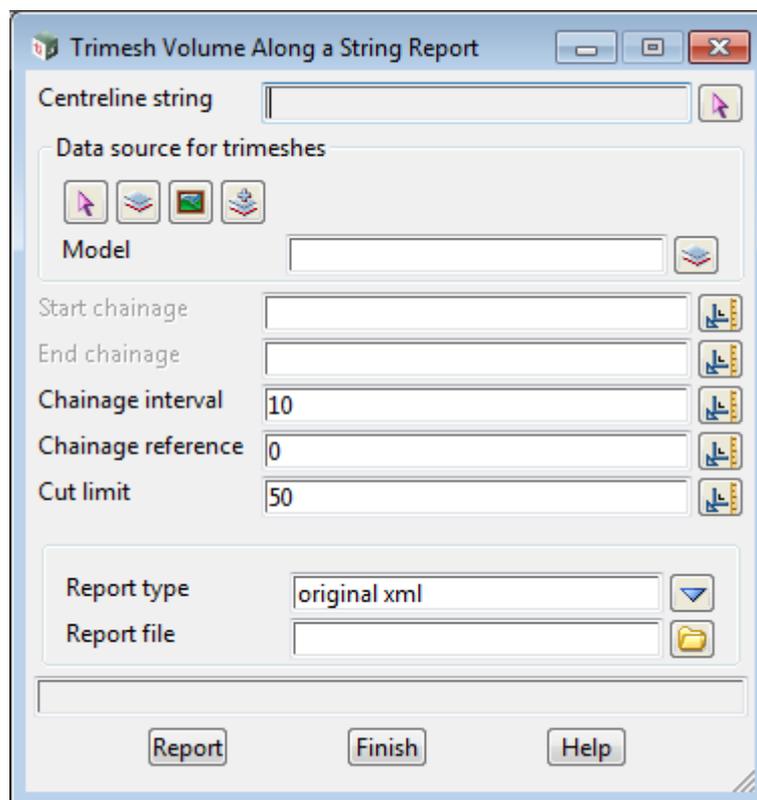
**Position of option on menu:** Design =>Volumes =>Trimesh =>Trimesh volumes along a string

This option calculates the volume of closed trimeshes between chainages on a selected string.

**Warning:** The volumes can only be calculated for trimeshes that in plan, the section line only cuts the trimesh twice. So in plan, the trimesh **can't** be a U or an S shape.



Selecting **Trimesh volumes along a string** report displays the **Trimesh Volumes Along a String Report** panel:



The fields and buttons used in this panel have the following functions.

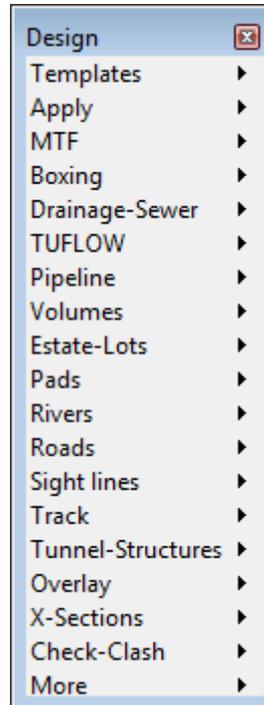
Field Description	Type	Defaults	Pop-Up
<b>Centreline string</b>	string select		

the string for defining the sections to be used in the volume calculations. The sections are taken at right angles to this string.

- Data source type** Model  
Data to search for trimeshes - for a full description go to [4.19.3 Data Source](#).
- Data source** input  
source of trimeshes to calculate the volumes along a string for:
- Start chainage** input  
if **not blank**, the chainage of the first section to use for volume calculations.  
If **blank**, start at the beginning of the selected string.
- End chainage** input  
if **not blank**, the chainage of the last section to use for volume calculations.  
If **blank**, end at the end of the selected string.
- Chainage interval** real box 10  
the chainage distance between the lines to section along.
- Cut limit** real box 50  
the plan distance that the section goes out from the selected string.
- Report type** choice box original xml, <Customize>  
output format for the report.  
For information on setting up custom reports from the generated XML file using xslts, see [4.30 Setting Up XML Reports](#).
- Report file** file box  
an XML file will be created and a report of this name, and of the type given by **Report type** will be generated from the XML file. If the file already exists, a panel with **Replace** and **Cancel** buttons will be displayed.
- Report** button  
the volume of each trimesh is calculated between each pair of sections and written to the XML file.

Return to [17.6 Trimesh](#).

# 22. Design Menu



The **Apply** and **MTF** options on the **Design** menu have been swapped around.

The **Apply** walk right menu has been rearranged, and **Sight lines** and **Check/clash** items added to the menu.

**MTF** will now stand for **Modifiers and Templates File** but that will be discussed in the Apply Menu section.

See

[22.1 Templates](#)

[22.2 Apply Menu](#)

[22.3 MTF](#)

[22.4 Boxing](#)

[22.4.5 Smart Chainages in Boxing Many Function](#)

[20 Drainage, Sewer and Tuflow](#)

[21 Volumes](#)

[22.5 Roads](#)

[22.6 Sight Lines](#)

[22.7 Tunnels and Structures](#)

[22.8 Check/Clash](#)

# 22.1 Templates

See

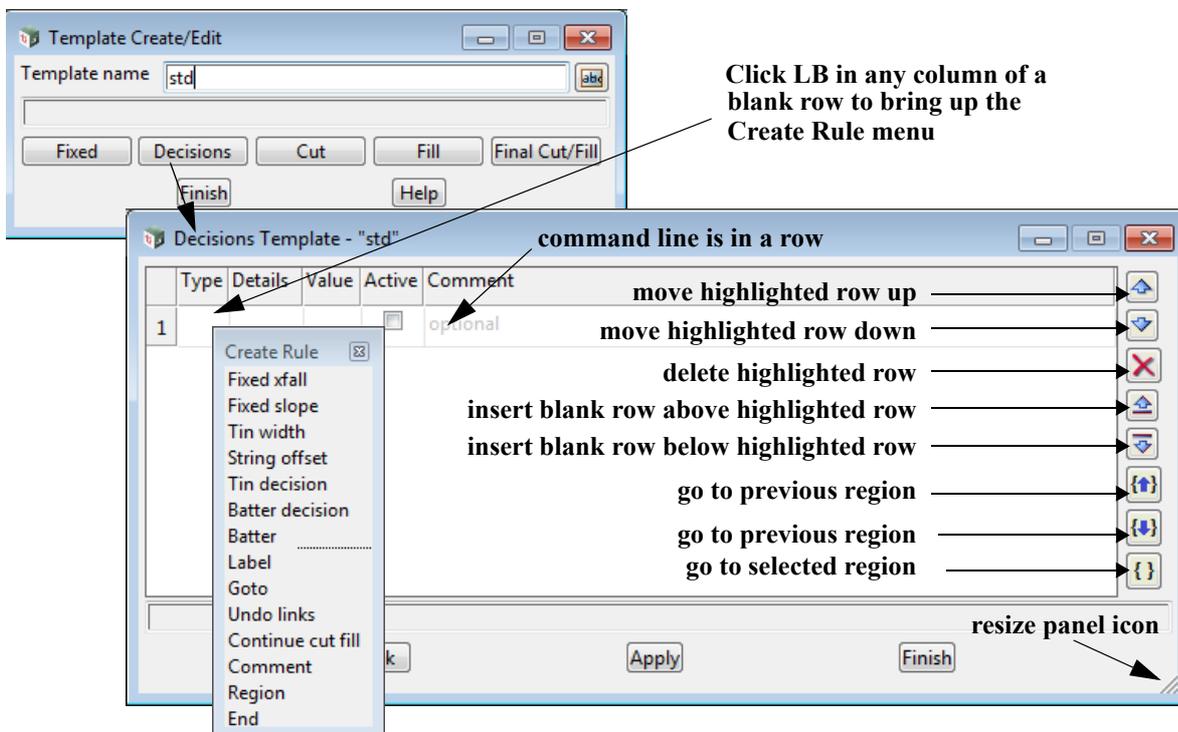
- [22.1.1 Decisions on Template Now a Grid](#)
- [22.1.2 Active Column in the Decisions Grid](#)
- [22.1.3 Regions in the Decisions Grid](#)

## 22.1.1 Decisions on Template Now a Grid

When *Creating* or *Editing* a **Template** with the option

Design =>Templates =>Create/edit

the **Decisions** button on the **Template Create/Edit** panel now brings up a grid.

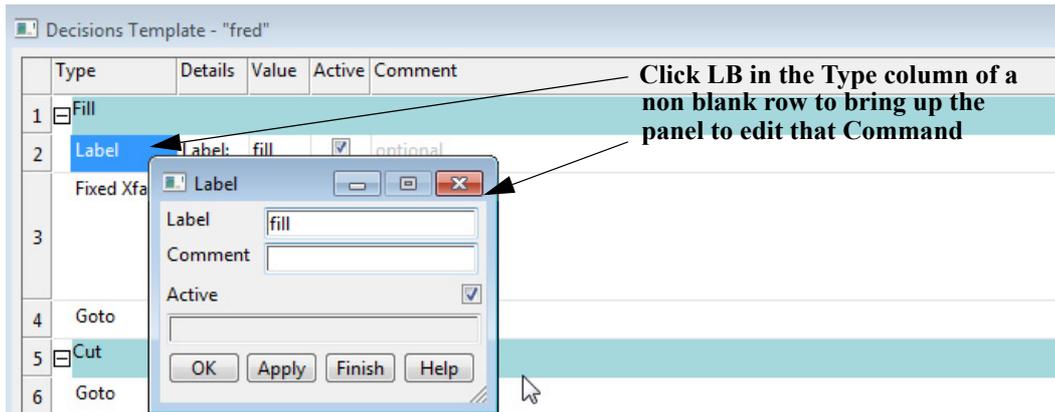


There are buttons on the right hand side to move rows **Up** and **Down**, **delete** a row, **insert** a blank row **Above/Below** the highlighted row.

The panel can be **resized** using the resize icon on the bottom left right hand corner of the panel.

Click LB in any highlighted column of a blank row brings up the **Create Rule** menu. Note this may involve two clicks - one to highlight a column in a row and the second click to bring up the **Create Rule** menu.

Click LB in any highlighted row in the **Type** column of a row already containing a Command brings up the Panel for that Command so any edits can be made. Note this may involve two clicks - one to highlight the **Type** column in a row and the second click to bring up the panel to edit that **Command**.



### 22.1.2 Active Column in the Decisions Grid

Each panel for the commands (rules) in Decisions now has an Active field and there is an corresponding Active column in the Decisions grid.

If the **Active** column for a command is ticked **on** then the command is used.

If the **Active** column for a command is not ticked, the command is not used.

### 22.1.3 Regions in the Decisions Grid

There is a new **Region** command (rule) for the **Decisions** grid.

A **Region** in a grid is a special command that has the property that a Region can be **collapsed**. And *collapsing* a Region means that all the rows after the Region command until the next Region command, are hidden in the grid. The row of the **Region** command is coloured light blue.

Regions must have a unique name within the Decisions grid.

See [12.11 Regions in Some Grids](#).

## 22.2 Apply Menu

MTF is now going to stand for **Modifiers and Templates File** rather than *Many Templates File*.

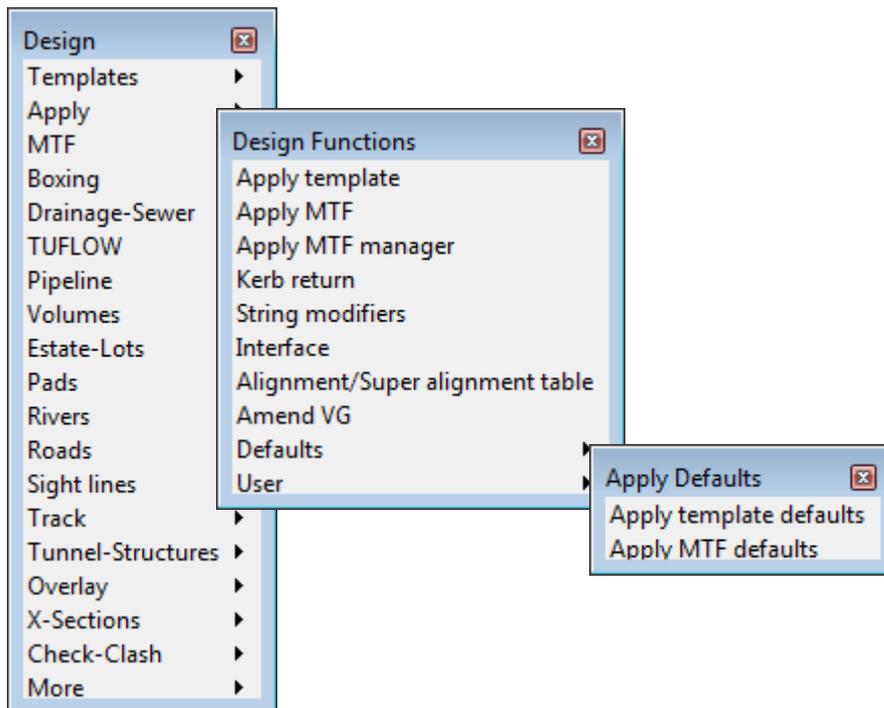
Also the **Apply** and the **Apply Many** will be referred to as **Apply Template** and **Apply MTF**.

This is to reflect the current usage where Templates as such may not even exist in an **Apply MTF** job.

It will take a while to eliminate the previous naming from all menus and panels especially as we don't want to effect existing Chains.

The *Apply defaults* have also been moved onto a Defaults walk-right menu.

So the walk-right on **Design =>Apply** will now look like:



See

[22.2.1 Apply MTF](#)

### 22.2.1 Apply MTF

See

[22.2.1.1 Smart Chainages in Apply MTF](#)

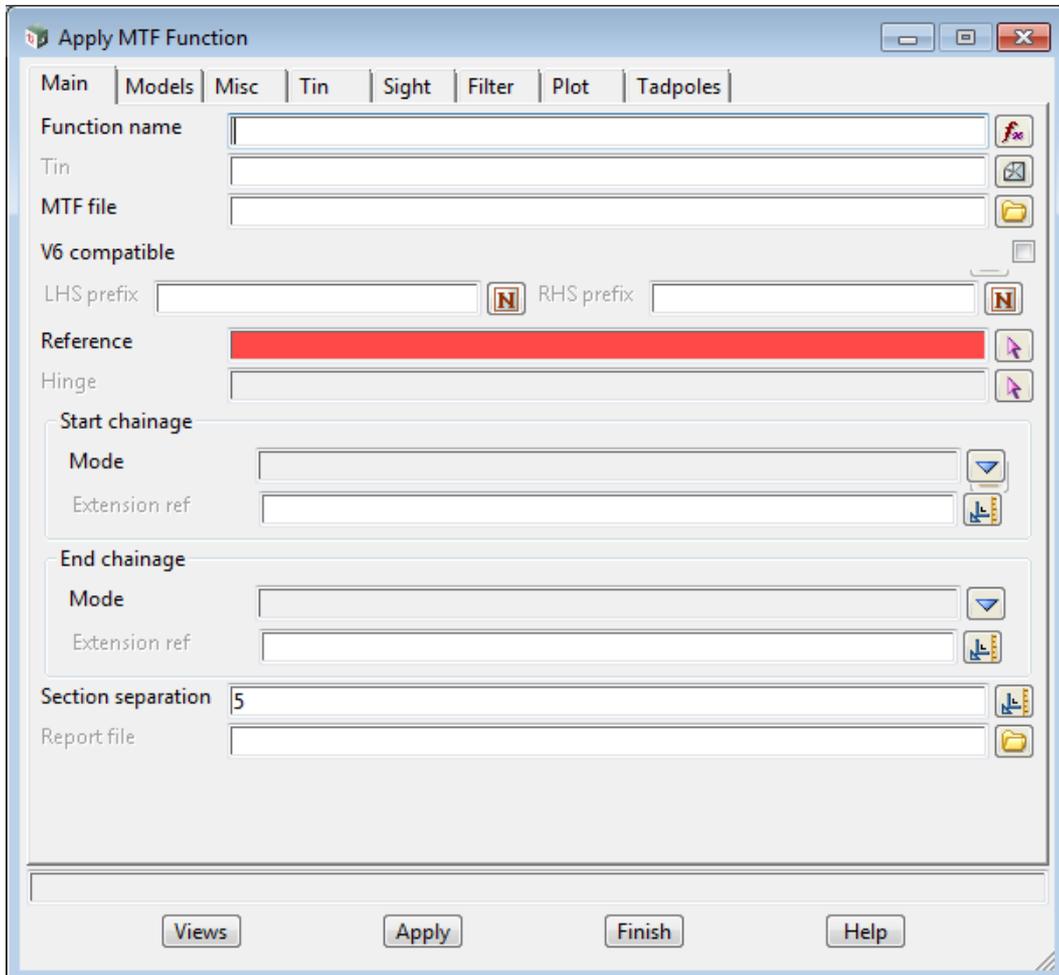
[22.2.1.1 Smart Chainages in Apply MTF](#)

[22.2.1.2 Summary Volumes Written to Output Window](#)

### 22.2.1.1 Smart Chainages in Apply MTF

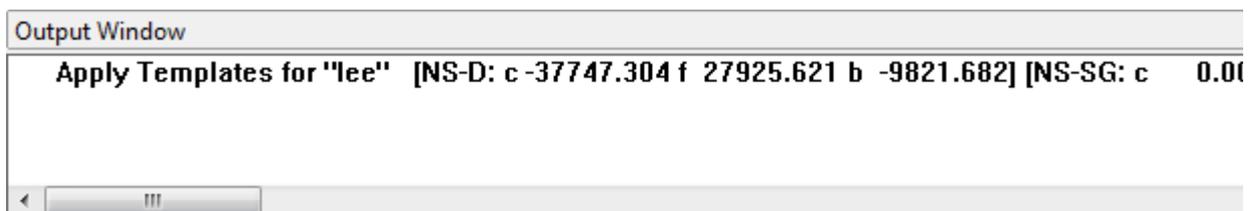
The **Apply MTF Template Function** now has **Smart Chainages** for its **Start Chainage** and **End Chainage**.

For the new information on **Smart Chainage**, see [9. Smart Chainages](#).



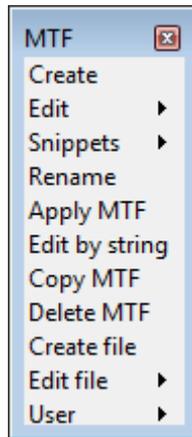
### 22.2.1.2 Summary Volumes Written to Output Window

When an **Apply MTF Template Function** now runs, the one line summary is written to the **Output Window**.



So you can have **Function results** turned off in the **Defaults** panel to stop the **Apply Templates for** panel from coming up with the summary results for each **Apply MTF** recalc (so you don't have to keep clicking on **OK**) but you still have a record of the results in the **Output Window**.

## 22.3 MTF

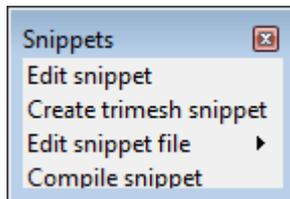


The **Create file** and **Edit file** options have been moved to just above **User** on the **MTF** menu. The **MTF Edit** has so many additions that it is in its own Chapter - see [23. MTF Edit](#);

See

[22.3.1 Snippets](#)

### 22.3.1 Snippets



See

[22.3.1.1 Edit Snippet](#)

[22.3.1.2 Create Trimesh Snippet](#)

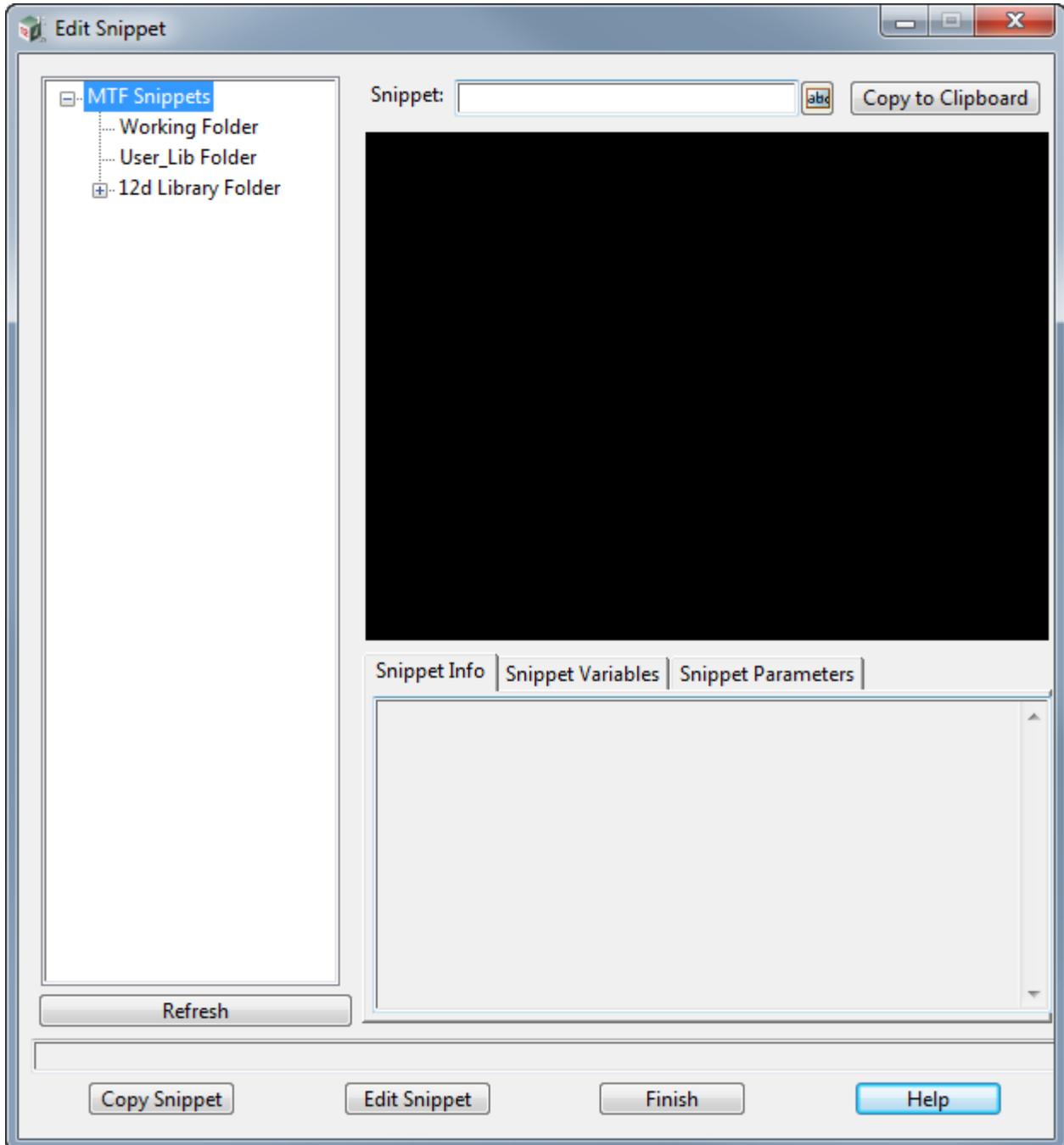
[22.3.1.3 Edit Snippet File](#)

[22.3.1.4 Compile Snippet](#)

### 22.3.1.1 Edit Snippet

**Position of option on menu:** Design =>MTF =>Snippet =>Edit snippet

Selecting **Edit snippet** brings up the **Edit Snippet** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>MTF Snippets</b>	Tree Box		

*lists all of the MTF Snippets available in the following folders:*

Working Folder:	MTF Snippets located in the current Working Folder
User_Lib Folder:	MTF Snippets located in the User Library (USER_LIB_4D)

12d Library Folder: MTF Snippets located in the 12d Model Library (LIB\_4D)  
(C:\Program Files (x86)\12d\12dmodel\11.00\Library, or  
C:\Program Files\12d\12dmodel\11.00\library depending on your setup)

*Expanding Working Folder, User\_Lib Folder of 12d Library Folder will list all the snippets in that folder. For information on the snippets in 12d Library Folder, see [22.3.1.5 12d Supplied Snippets](#).*

*Clicking on a snippet name in the list displays the following information about the selected snippet:*

**Snippet** Input box

*displays the file path and name of the selected MTF Snippet.*

**Copy to Clipboard** Button

*copies the selected MTF Snippet file path to the clipboard*

*This may be pasted into the **Modifiers->MTF Snippet** panel, or useful for creating snippets that call other snippets.*

**MTF Snippet Preview** Draw box

*displays a preview image of the selected MTF Snippet.*

*Users can create their own preview images using the following guidelines:*

- *A preview image must have the same name as its associated MTF Snippet. For example; a snippet named "My\_Snippet.MTFSNIPPET", will have a preview image named "My\_Snippet.jpg",*
- *The preview image should have the dimensions 400 pixels wide by 300 pixels high,*
- *The preview image must be in jpeg (\*.jpg) image format,*
- *The preview image must be located in either your "\$USER/images" folder (USER\_4D) or your current Working Folder.*

**Refresh** Button

*updates the MTF Snippets tree box.*

*When clicked, the macro will find and display all MTF Snippets in the current project Working Folder, User Library Folder, and 12d Library Folder.*

*Note: To fully refresh a list, the relevant tree must also be collapsed and then re-expanded.*

## Snippet Info Tab

**Snippet Info** Text Edit Box

*displays information about the currently selected MTF Snippet.*

*Users can display their own custom information for a snippet by using the following guidelines:*

- *Open the MTF Snippet in a text editor, by select Edit Snippet,*
- *At the top of the file, prefix your information with: // INFO*

*For example; if your snippet file contains:*

*// INFO Snippet Type: INSERT*

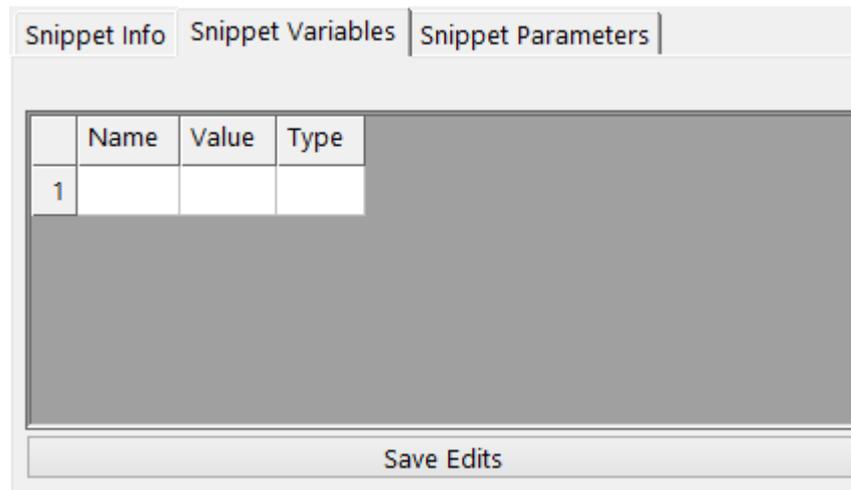
*// INFO Description: Kerb & Gutter*

*The **Snippet Info** text box will display:*

*Snippet Type: INSERT*

*Description: Kerb & Gutter*

## Snippet Variables Tab



### Snippet Variables Grid Control box

lists the currently selected MTF Snippet variables.

Each row represents a variable.

Column 1 - represents the variable name,

Column 2 - represents the variable value,

Column 3 - represents the variable type.

MTF Snippet variable types include: Number, Text

Only variables that define a single value are displayed (i.e.; formulas are ignored).

For example:

#define	_A	1.23
@ def_tok	A	1.23
@ def_tok	STR1	KI

Users can create new variables, edit existing variables and delete existing variables.

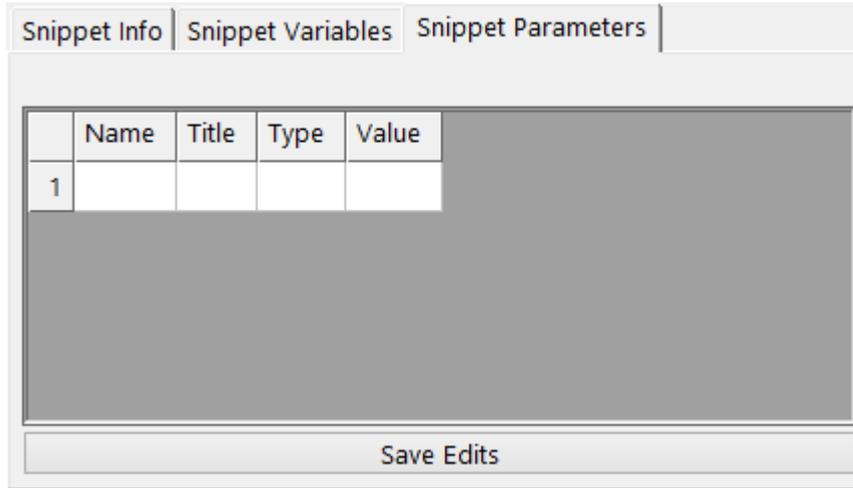
Note: Creating and/or deleting a variable in the Grid Control Box does not change how that variable is used elsewhere in the snippet.

### Save Edits button

open the **Unsaved Changes** panel to write out the **Snippet Variables** (Grid Control Box) to the MTF Snippet file.

Note: 12d Model Library snippets and User Library snippets cannot be edited via the panel. Therefore this button is disabled when these snippets are selected.

## Snippet Parameters Tab



### Snippet Parameters Grid Control box

*lists the currently selected MTF Snippet parameters.*

*Each row represents a parameter.*

*Column 1 - represents the parameter name,*

*Column 2 - represents the parameter title,*

*Column 3 - represents the parameter type,*

*Column 4 - represents the parameter value.*

*Parameters to be displayed in the Edit Snippet panel must be defined in the snippet\_params.4d file.*

*Only parameters of the following types may be displayed and edited: TEXT, REAL, INTEGER*

*Users can create new parameters, edit existing parameters and delete existing parameters.*

*Note: Creating and/or deleting a parameter in the Grid Control Box does not change how that parameter is used elsewhere in the snippet.*

### Save Edits button

*opens the **Unsaved Changes** panel to write out the **Snippet Parameters** (Grid Control Box) to the MTF Snippet file.*

*Note: 12d Model Library snippets and User Library snippets cannot be edited via the panel. Therefore this button is disabled when these snippets are selected.*

### Copy Snippet button

*opens the **Copy Snippet To** panel and allows the user to copy the selected snippet to the current Working Folder.*

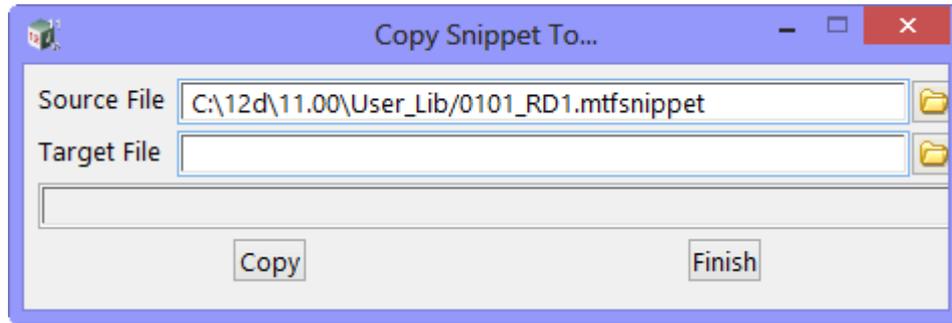
### Edit Snippet button

*open the selected snippet in a text editor.*

*The default text editor is Notepad, unless a text editor is defined in the users env.4d:EDITOR\_4D*

*Note: The text editors Multiple Instance settings will control whether snippets open for editing in the same window or create a new window.*

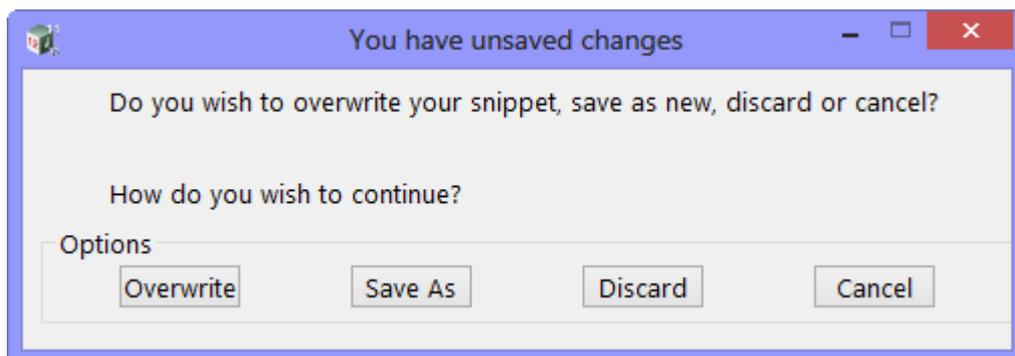
Selecting the **Copy Snippet** button brings up the **Copy Snippet To...** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Source File</b>	File Box		
<i>file path and name of the MTF Snippet being copied.</i>			
<i>When the <b>Copy Snippet To</b> panel opens, this field is auto-populated with the currently selected MTF Snippet from the MTF Snippets tree.</i>			
<b>Target File</b>	File Box		
<i>enter a new file name and path for the MTF Snippet being copied.</i>			
<i>Entering a file name with no path will copy the MTF Snippet to the current Working Folder.</i>			
<b>Copy</b>	Button		
<i>when clicked the <b>Source File</b> is copied with the <b>Target File</b> path and name.</i>			

Selecting the **Save Edits** button brings up the **Unsaved Changes** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Overwrite</b>	Button		
<i>writes all variables in the <b>MTF Variables</b> Grid Control Box to the currently selected MTF Snippet file.</i>			
<i>Existing variables of the same name will be overwritten.</i>			
<b>Save As</b>	Button		

*save all variables in the **MTF Variables** Grid Control Box to a new MTF Snippet file.*

*Selecting this button brings up the **Copy Snippet To** panel.*

**Discard** Button

*all edits made to the variables in the **MTF Variables** Grid Control Box are discarded and the currently selected MTF Snippet file is not edited.*

**Cancel** Button

*closes the **Unsaved Changes** panel.*

### **How snippet\_params.4d Works**

snippet\_params.4d is a file that allows users to control the visibility of MTF Snippet Parameters in the Snippet Parameters Grid\_CtrlBox.

File Format:

- *Plain Text*
- *Each line should contain one MTF Snippet Parameter name. For example: LANE1*
- *MTF Snippet Parameter names are case sensitive*
- *Comments are allowed in the file, and should begin with: //*

### **Where should snippet\_params.4d be located?**

By default the file is located in:

C:\Program Files\12d\12dmodel\11.00\set\_ups

or

C:\Program Files (x86)\12d\12dmodel\11.00\set\_ups

Users can copy this file to their User directory (typically C:\12d\11.00\User) or their current 12d Model project directory. The macro will find and read this file in those directories. The macros search order is:

- *Search the current 12d Model project directory for Snippet\_Params.4d. If found; read.*
- *Search the User directory for Snippet\_Params.4d. If found; read.*
- *Search the Set\_Ups directory for Snippet\_Params.4d. If found; read.*

### 22.3.1.2 Create Trimesh Snippet

**Position of option on menu:** Design =>MTF =>Snippet =>Create trimesh snippet

Not yet documented.

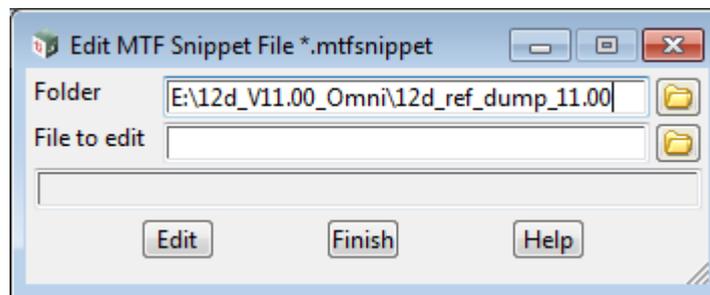
### 22.3.1.3 Edit Snippet File

**Position of option on menu:** Design =>MTF =>Snippet =>Edit snippet file

The **Edit snippet file** option is used to edit MTF Snippet files (\*.mtfsnippet) with the text editor pointed to by the EDIT\_4D environment variable.

The **Edit snippet file** option has two modes of operation - clicking on the option **Edit snippet file** itself, or by activating the **MTF =>Snippets =>Edit snippet file** option's walk-right menu, **folder \*.mtfsnippet**.

Selecting **Edit snippet file** itself brings up the **Edit MTF File \*.mtfsnippet** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Folder</b> <i>name of the folder for the .mtf snippet file.</i>	folder box	current folder	
<b>File to edit</b> <i>name of the MTF Snippet file, in Folder, to edit.</i>	file box		*.mtfsnippet files
<b>Edit</b> <i>edit the MTF Snippet file given by the Folder and File to edit panel fields by the text editor pointed to by the EDIT_4D environment variable. If the file given in the File to edit field does not exist, then a new file is created which already has each of the section headers set up.</i>	button		

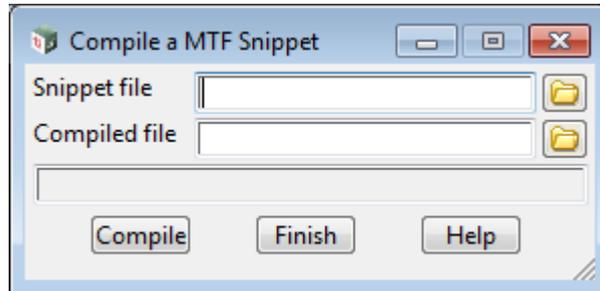
Similarly the **Edit snippet file** walk-right menu provides a list all the MTF Snippet files (files ending in .mtfsnippet) in the current folder. When a file is selected from the list, it is automatically loaded into the text editor.

### 22.3.1.4 Compile Snippet

**Position of option on menu:** Design =>MTF =>Snippet =>Compile snippet

The **Compile snippet** option is used to create a compiled version of a Snippet file. The compiled snippet file can **not** be read with a text editor.

Selecting **Compile snippet** itself brings up the **Edit MTF File \*.mtfsnippet** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet file</b> <i>name of the MTF Snippet file to compile.</i>	file box		*.mtfsnippet files
<b>Compiled file</b> <i>name of the compiled MTF Snippet file.</i>	file box		*.mtfsnippetc files
<b>Compile</b> <i>compile the snippet given in the <b>Snippet file</b> field and write the compiled snippet out with the name given in the <b>Compile file</b> field.</i>	button		

### 22.3.1.5 12d Supplied Snippets

See

[22.3.1.5.1 TRI 1PT TRENCH DES\\_LAY](#)

[22.3.1.5.2 TRI 1PT TRENCH EXT\\_STRS](#)

[22.3.1.5.3 TRI 2PT PAV DES\\_LAY](#)

[22.3.1.5.4 TRI 2PT PAV EXT\\_STRS](#)

[22.3.1.5.5 TRI 2PT PAV NAMED GRADES](#)

[22.3.1.5.6 TRI 2PT PAV NAMED GRADES EXT\\_STRS](#)

[22.3.1.5.7 TRI 2PT PAV STOP NAMED GRADES](#)

[22.3.1.5.8 TRI 2PT PAV STOP NAMED GRADES EXT\\_STRS](#)

[22.3.1.5.9 TRI 2PT STRUCT\\_FILL](#)

[22.3.1.5.10 TRI 3PT PAV DES\\_LAY](#)

[22.3.1.5.11 TRI 3PT PAV EXT\\_STRS](#)

[22.3.1.5.12 TRI KERB\\_PROFILE](#)

### 22.3.1.5.1 TRI\_1PT\_TRENCH\_DES\_LAY

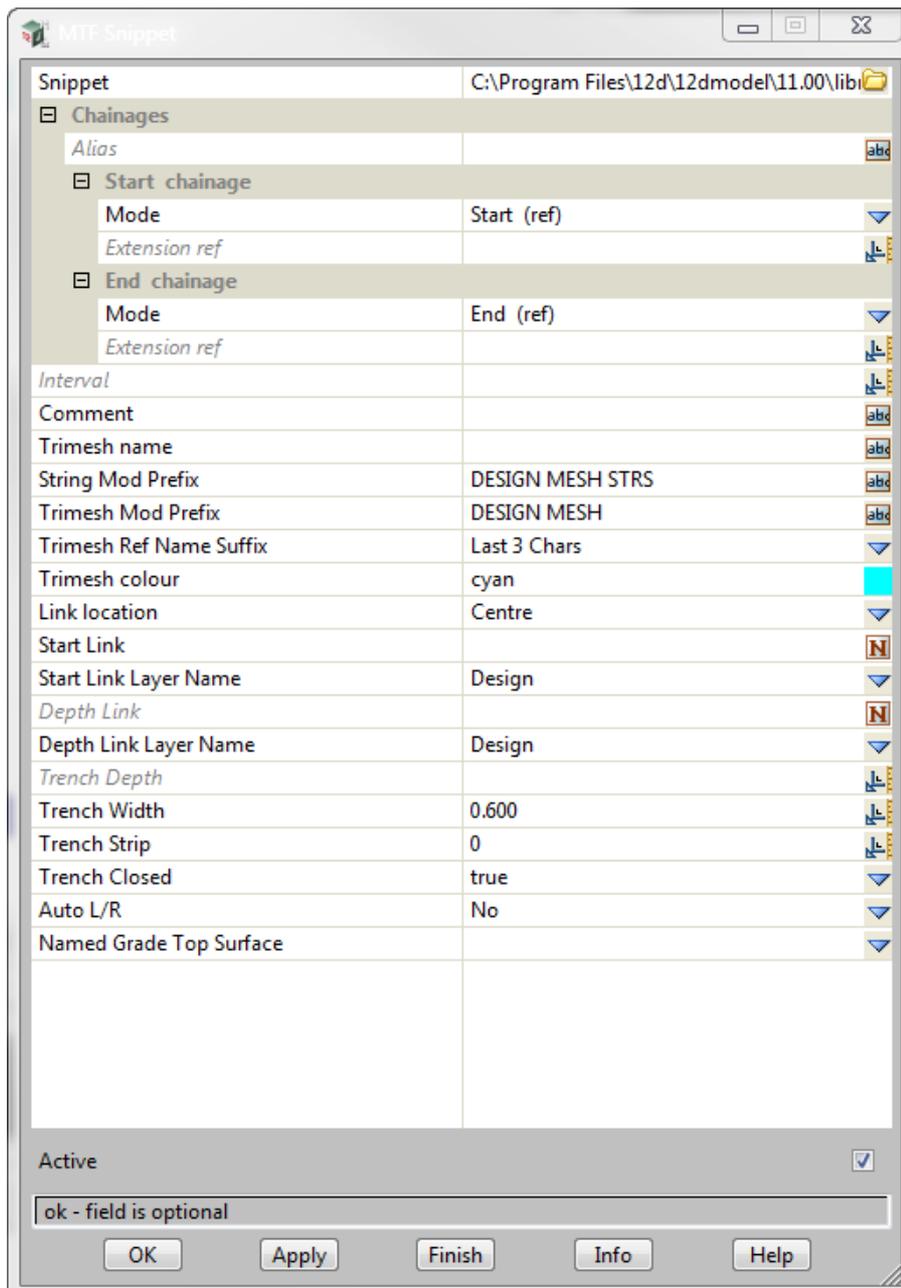
This Snippet is used to create a trench pavement, using <Design> or <Layer> strings created in your MTF.

It is linked to the Start Link at a specified location of left, centre or right.

A depth can be specified either to another link or a set value.

The top surface of the trench is defined by a Named Grade.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>MTF file</b>	file box		*.mtf files

*name of the MTF file to create.*

**Snippet** snippet box \*.mtfsnippet, \*.mtfsnippetc files  
*snippet to run.*

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*for information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Trimesh Name** input  
*trimesh name that is used in model names below*

**String Mod Prefix** input DESIGN MESH STRS

**Trimesh Mod Prefix** input DESIGN MESH

**Trimesh Ref Name Suffix** choice box Last 3 Chars All Chars  
 Last Char  
 Last 2 Chars  
 Last 3 Chars  
 Last 4 Chars  
 None

*(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)*

**Trimesh Colour** colour box cyan various

**Link Location** choice box Centre Centre  
 LHS  
 RHS

**Start Link** name

**Start Link Layer Name** choice box Design various

**Depth Link** name

**Depth Link Layer Name** choice box Design various

*(Another link can be used to control the depth of the trench.*

*It is optional so if not set then the following Trench depth is used)*

**Trench Depth** real

*(This is optional so if not set then the Depth Link above is used)*

**Trench Width** real 0.6

**Trench Strip** real 0

*(Allows a strip depth measured from the top of the trench)*

**Trench Closed** choice box true true / false

**Auto L/R** choice box No Yes / No

if set to <No> then the Start and End Links remain unchanged.

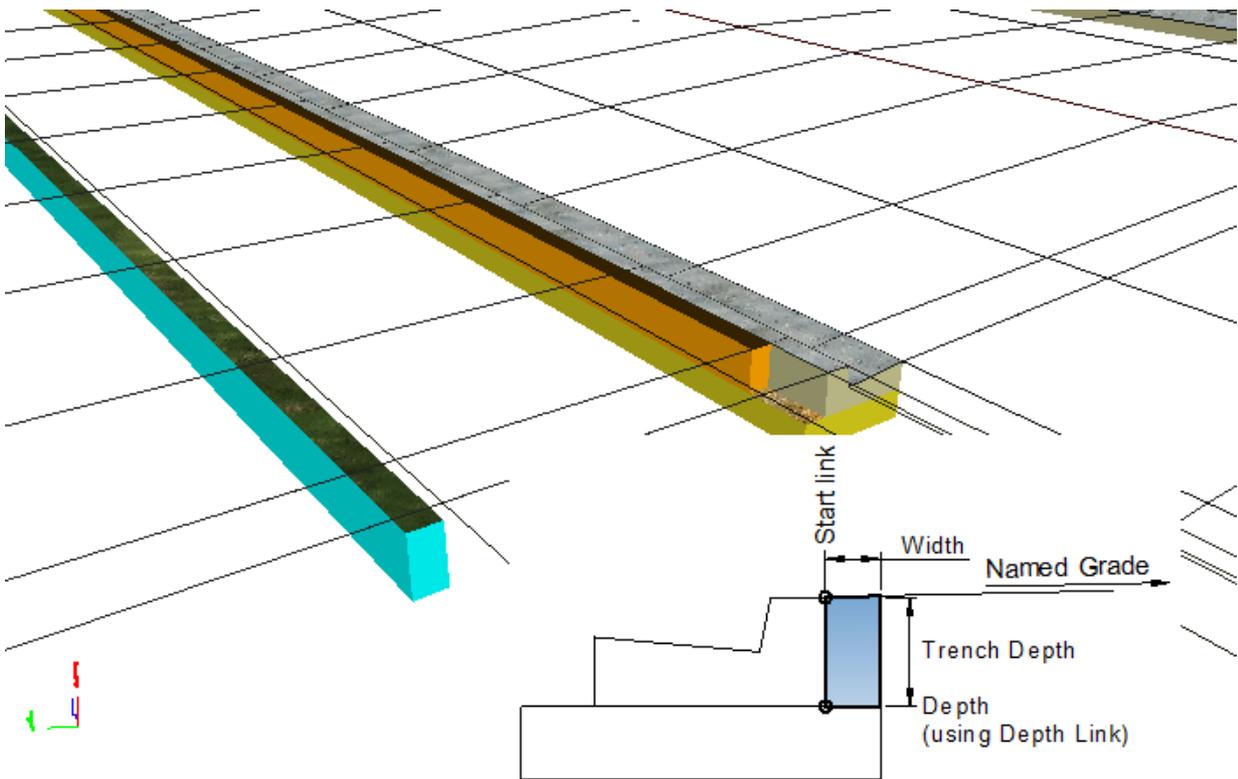
If set to <Yes> then the Start and End Links need the suffix for L/R removed.

e.g. Any <Design> string names like **ESL** would need the <L> removed

e.g. Any <Layer> string names like **PVBALRO1** would need the <LRO1> removed

**Named Grade Top Surface** choice box

(The top of the trench will follow this grade while the bottom of the trench is horizontal)



Continue to [22.3.1.5.2 TRI\\_1PT\\_TRENCH\\_EXT\\_STRS](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

### 22.3.1.5.2 TRI\_1PT\_TRENCH\_EXT\_STRS

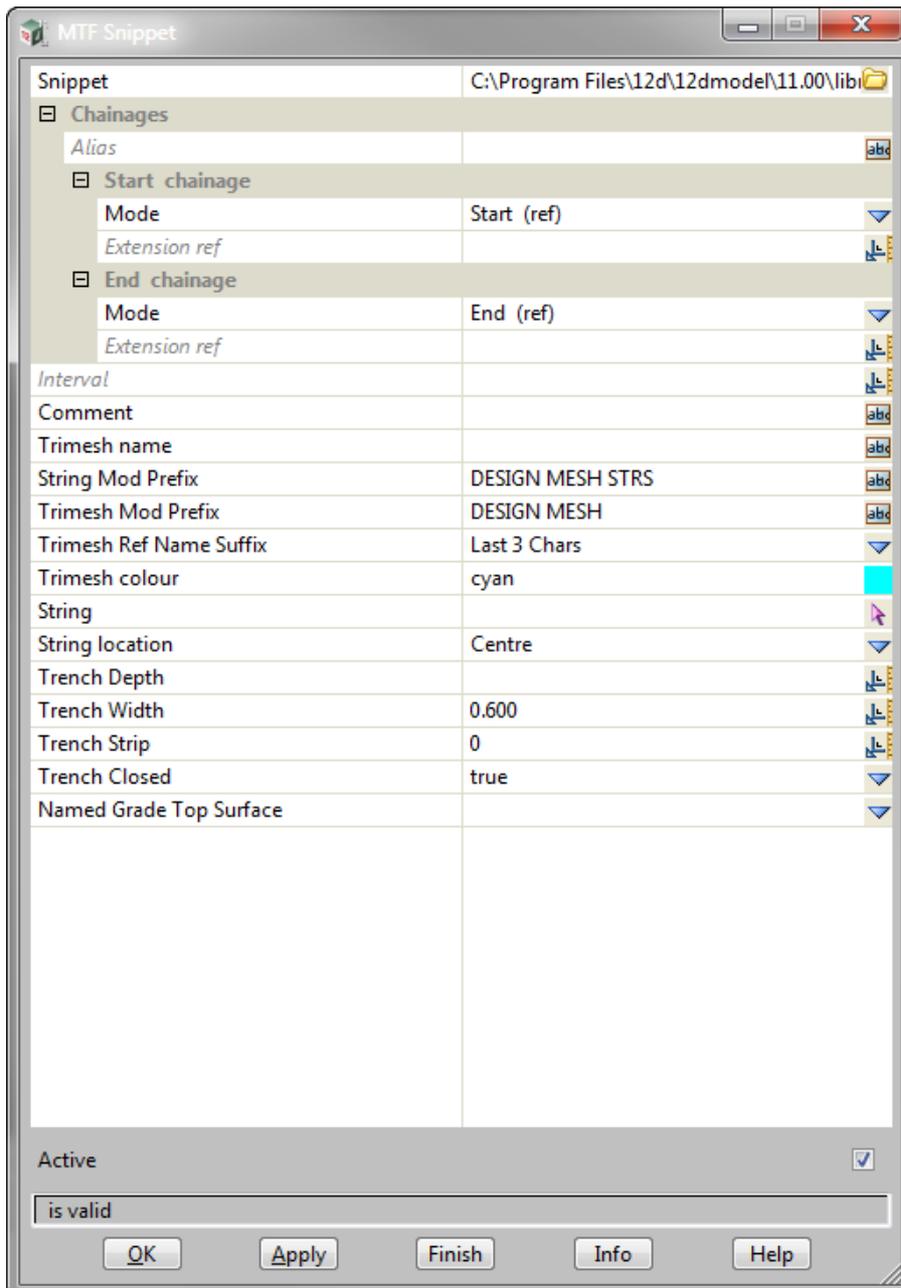
This Snippet is used to create a trench pavement, using External strings NOT created in your MTF.

It is linked to the Start Link at a specified location of left, centre or right.

A depth can be specified either to another link or a set value.

The top surface of the trench is defined by a Named Grade

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

*snippet to run.*

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

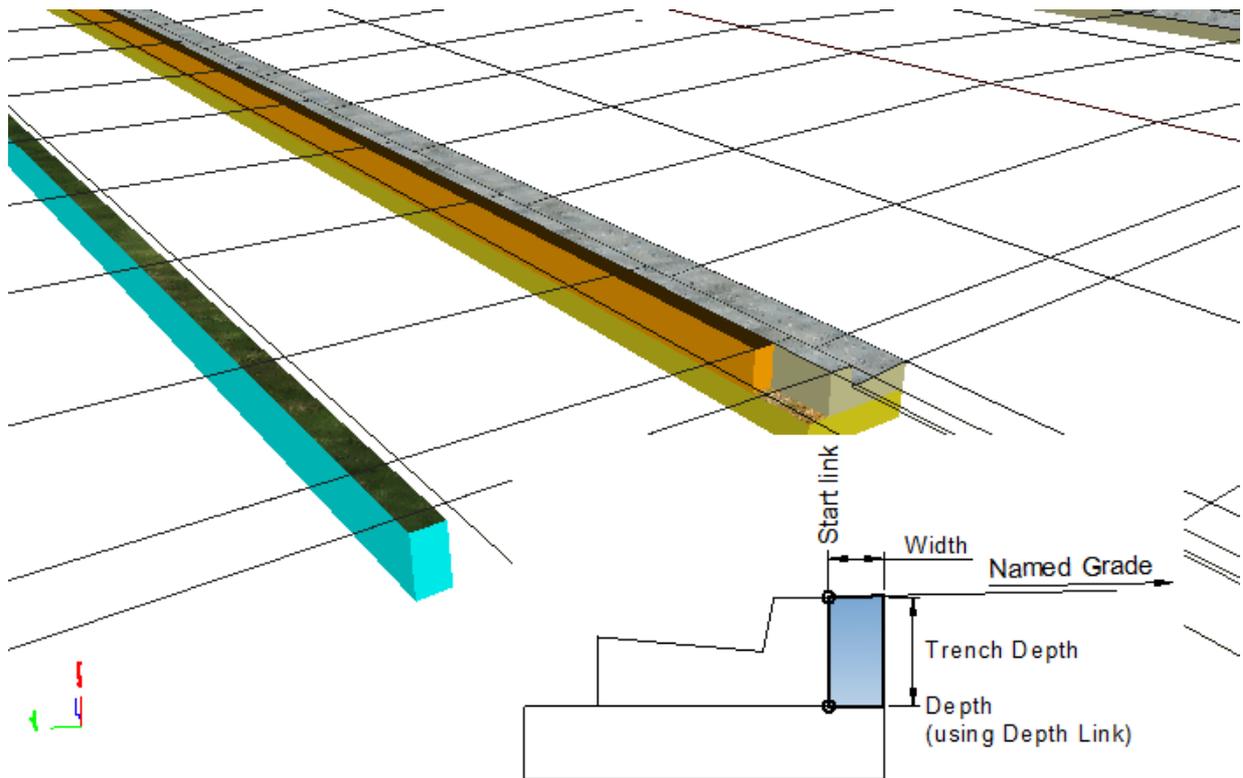
*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh RefName Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>String</b>	select		
<b>String Location</b>	choice box	Centre	Centre LHS RHS
<b>Trench Depth</b>	real		
<b>Trench Width</b>	real		0.6
<b>Trench Strip</b>	real		0
	<i>(Allows a strip depth measured from the top of the trench)</i>		
<b>Trench Closed</b>	choice box	true	true / false
<b>Named Grade Top Surface</b>	choice box		

*(The top of the trench will follow this grade while the bottom of the trench is horizontal)*



Continue to [22.3.1.5.3 TRI\\_2PT\\_PAV\\_DES\\_LAY](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

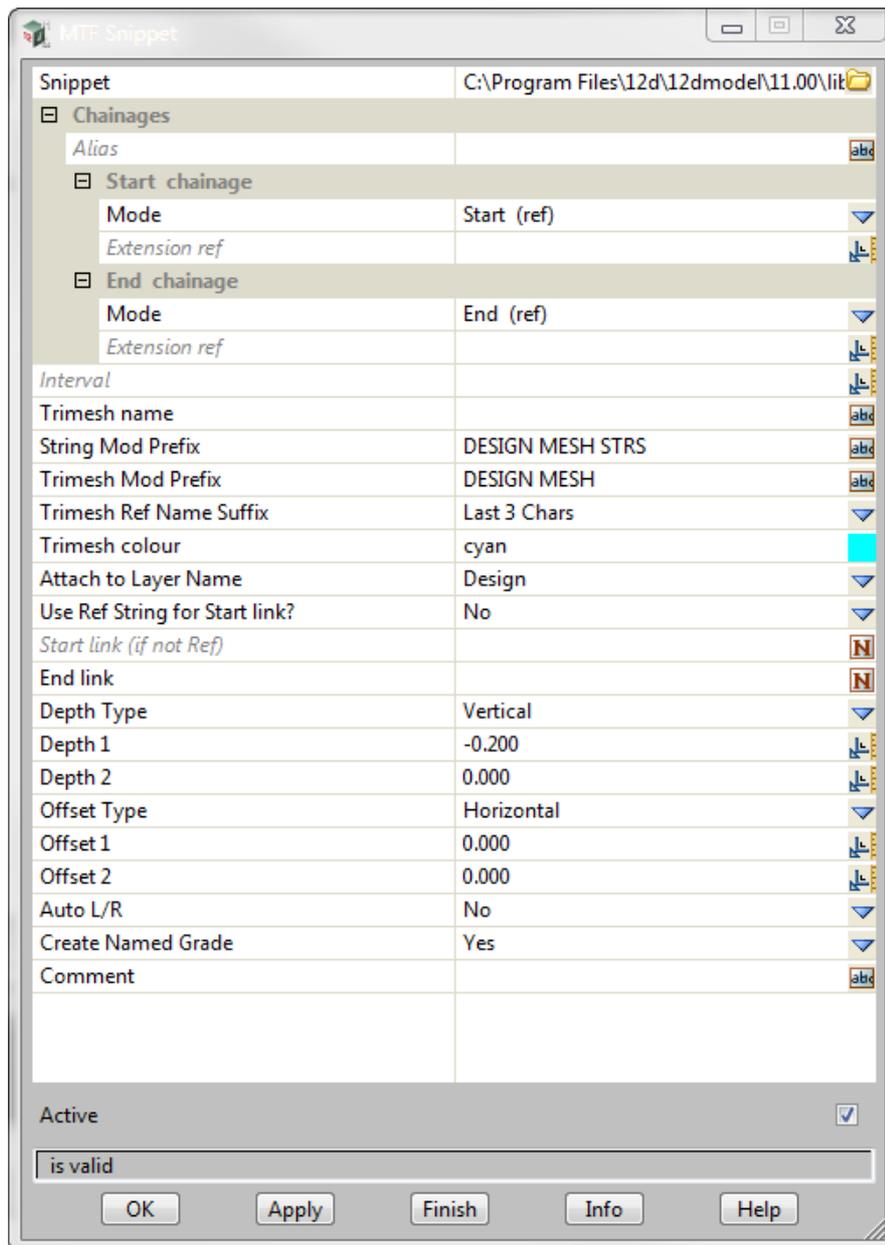
### 22.3.1.5.3 TRI\_2PT\_PAV\_DES\_LAY

This Snippet is used to create typical pavement, using <Design> or <Layer> strings created in your MTF.

It is linked to the Start and End Links at a depth specified.

Offsets from these links can be set and measured Horizontally or down the slope created from the two links.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None

*(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)*

<b>Trimesh Colour</b>			colour boxcyanvarious
<b>Attach to Layer Name</b>	choice box	Design	various
<b>Use Ref String for Start Link?</b>	choice box	No	Yes / No
<b>Start Link (if not Ref)</b>	name		
	<i>(Optional if use ref string is set to Yes)</i>		
<b>End Link</b>	name		
<b>Depth Type</b>	choice box	Vertical	Vertical Normal Top Surface Bottom Surface
<b>Depth 1</b>			real-0.2
<b>Depth 2</b>			real
	<i>(Optional...if not used Depth 1 is set throughout)</i>		
<b>Offset Type</b>	choice box	Horizontal	Horizontal On Slope
<b>Offset 1</b>	real		0
<b>Offset 2</b>	real		0
<b>Auto L/R</b>	choice box	No	Yes / No

*if set to <No> then the Start and End Links remain unchanged.*

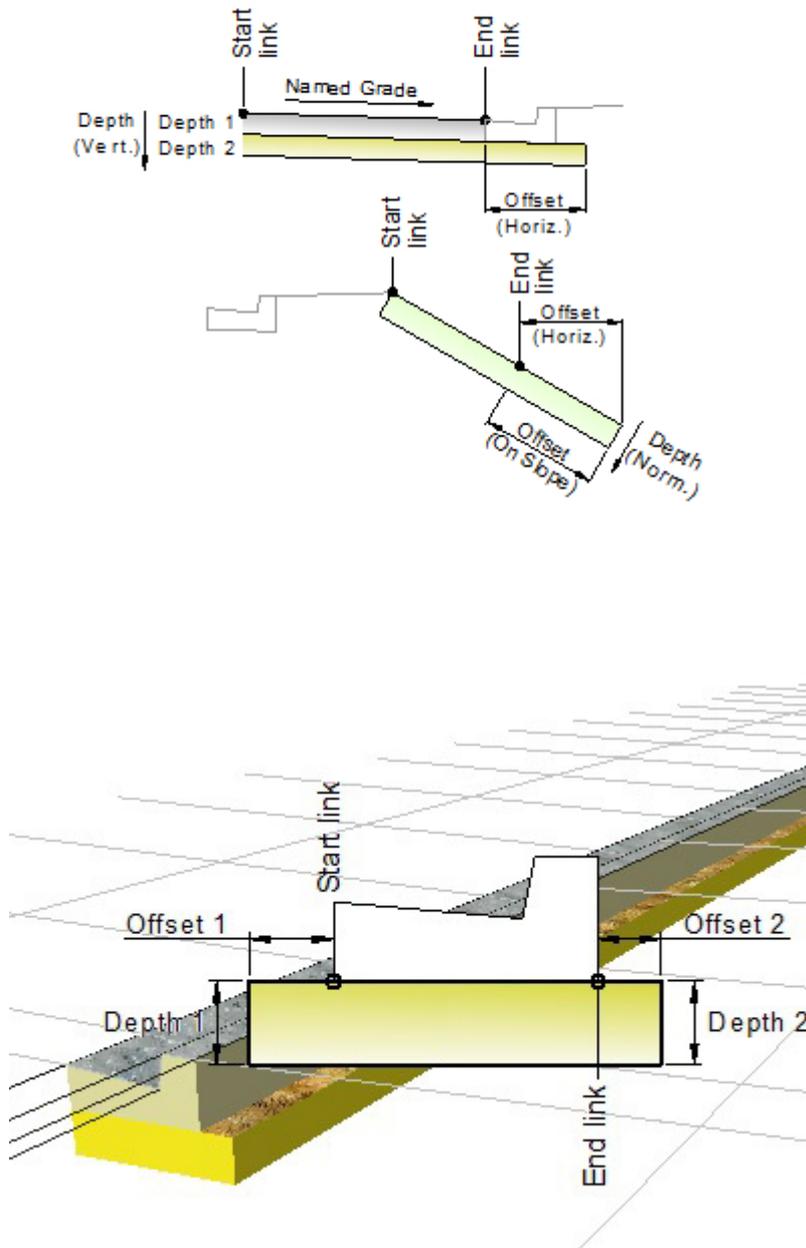
*If set to <Yes> then the Start and End Links need the suffix for L/R removed.*

e.g Any <Design> string names like **ESL** would need the <L> removed

e.g Any <Layer> string names like **PVBALROI** would need the <LROI> removed

**Create Named Grades**      choice box      No      Yes / No

if set to <Yes> then a Named Grade in the format <Start Link Name \_ End Link Name> is created for use in the MTF by other modifiers



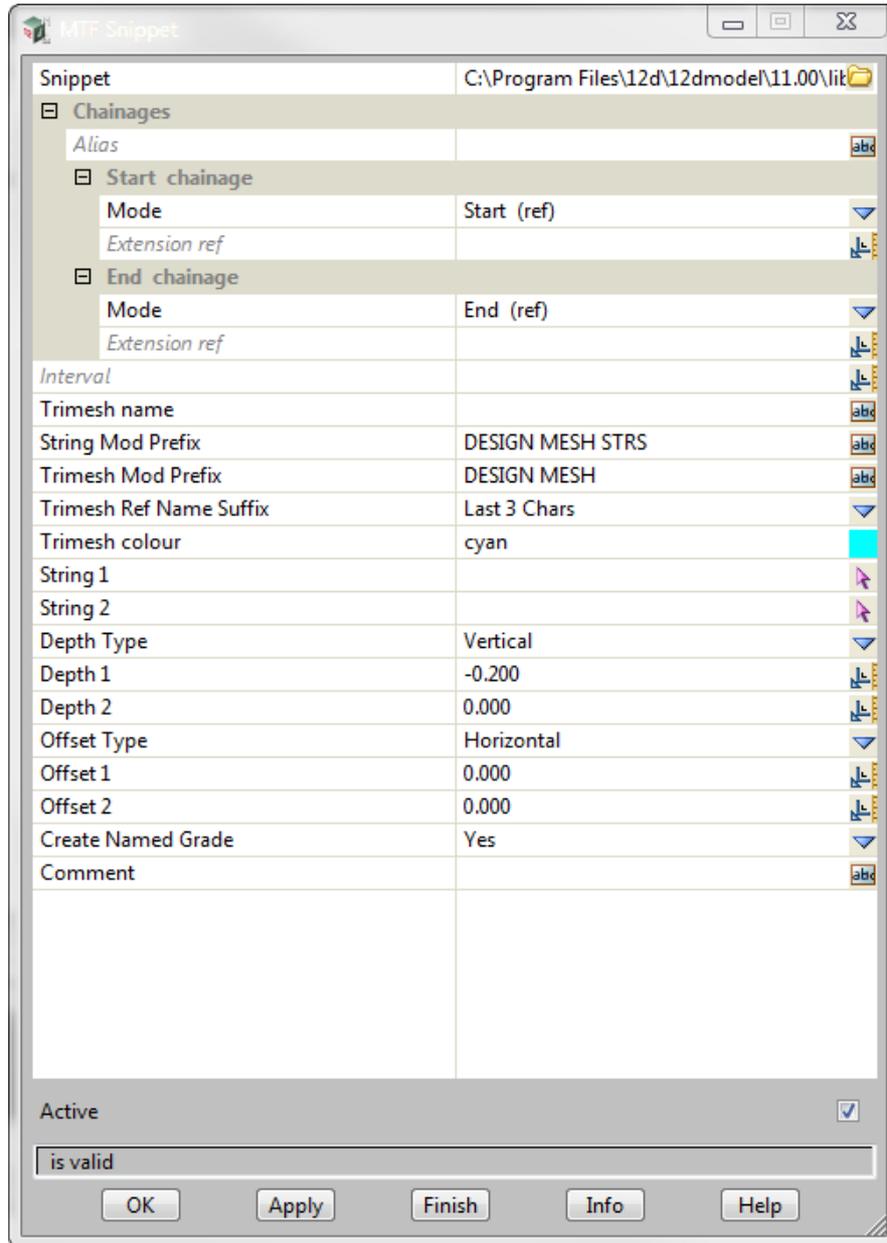
Continue to [22.3.1.5.4 TRI\\_2PT\\_PAV\\_EXT\\_STRS](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

### 22.3.1.5.4 TRI\_2PT\_PAV\_EXT\_STRS

This Snippet is used to create typical pavement, using external strings NOT created in your MTF. It is linked to the String 1 and String 2 at a depth specified.

Offsets from these links can be set and measured Horizontally or down the slope created from the two links.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

**Alias, Start Chainage, End Chainage, Interval**

defines the start and end chainages to apply a snippet.

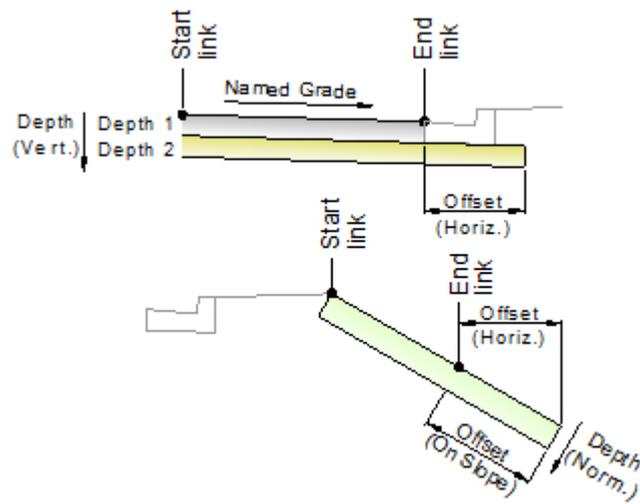
For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

**Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>Attach to Layer Name</b>	choice box	Design	various
<b>Use Ref String for Start Link?</b>	choice box	No	Yes / No
<b>String 1</b>	select		
<b>String 2</b>	select		
<b>Depth Type</b>	choice box	Vertical	Vertical Normal Top Surface Bottom Surface
<b>Depth 1</b>	real		-0.2
<b>Depth 2</b>	real		
	<i>(Optional...if not used Depth 1 is set throughout)</i>		
<b>Offset Type</b>	choice box	Horizontal	Horizontal On Slope
<b>Offset 1</b>			real0
<b>Offset 2</b>			real0
<b>Create Named Grades</b>	choice box	No	Yes / No

If set to <Yes> then a Named Grade in the format <String 1 \_ String 2> is created for use in the MTF by other modifiers.



Continue to [22.3.1.5.5 TRI\\_2PT\\_PAV\\_NAMED\\_GRADES](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#) .

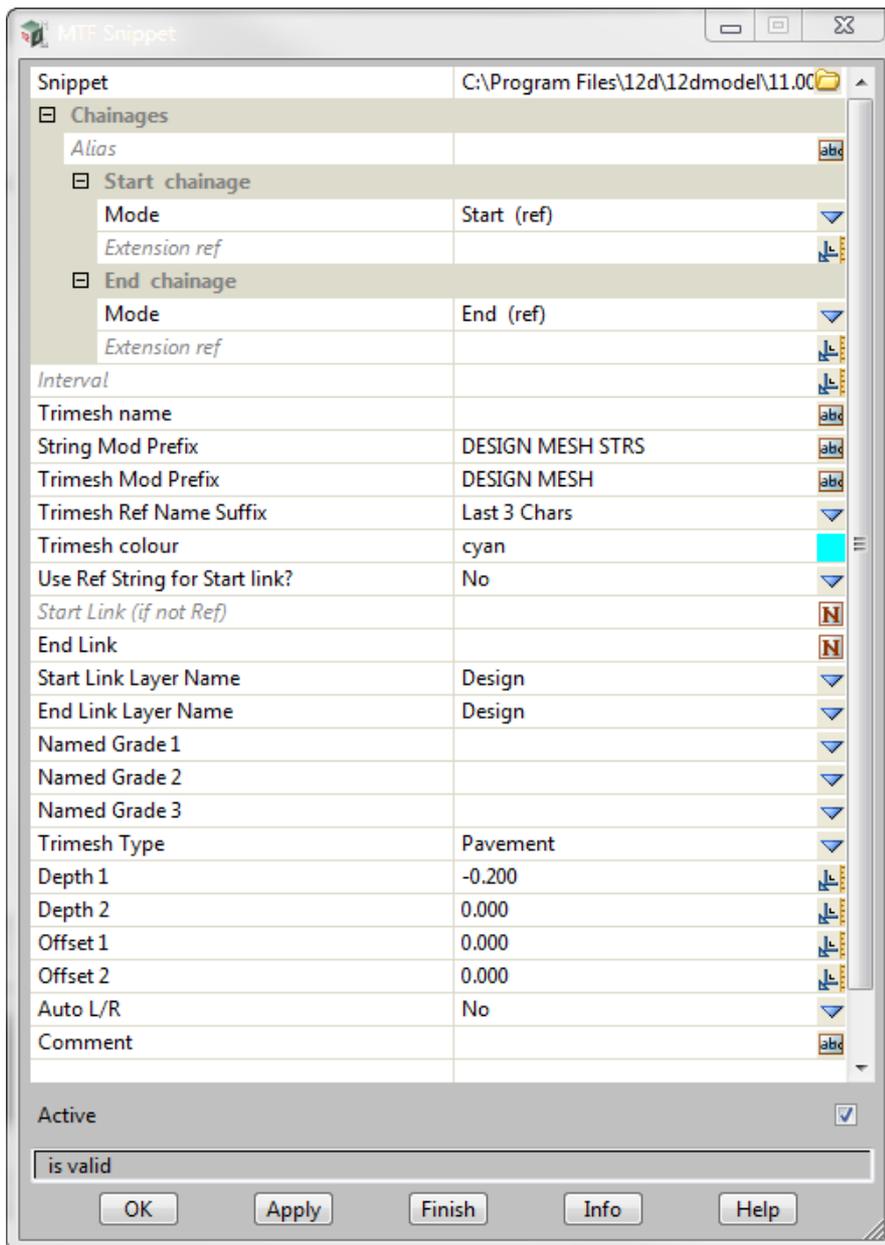
### 22.3.1.5.5 TRI\_2PT\_PAV\_NAMED\_GRADES

This Snippet is used to create typical pavement, using <Design> or <Layer> strings created in your MTF.

It is linked to the Start and End Links at a depth specified.

Offsets from these links can be set and measured Horizontally or down the slope created from the two links

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files



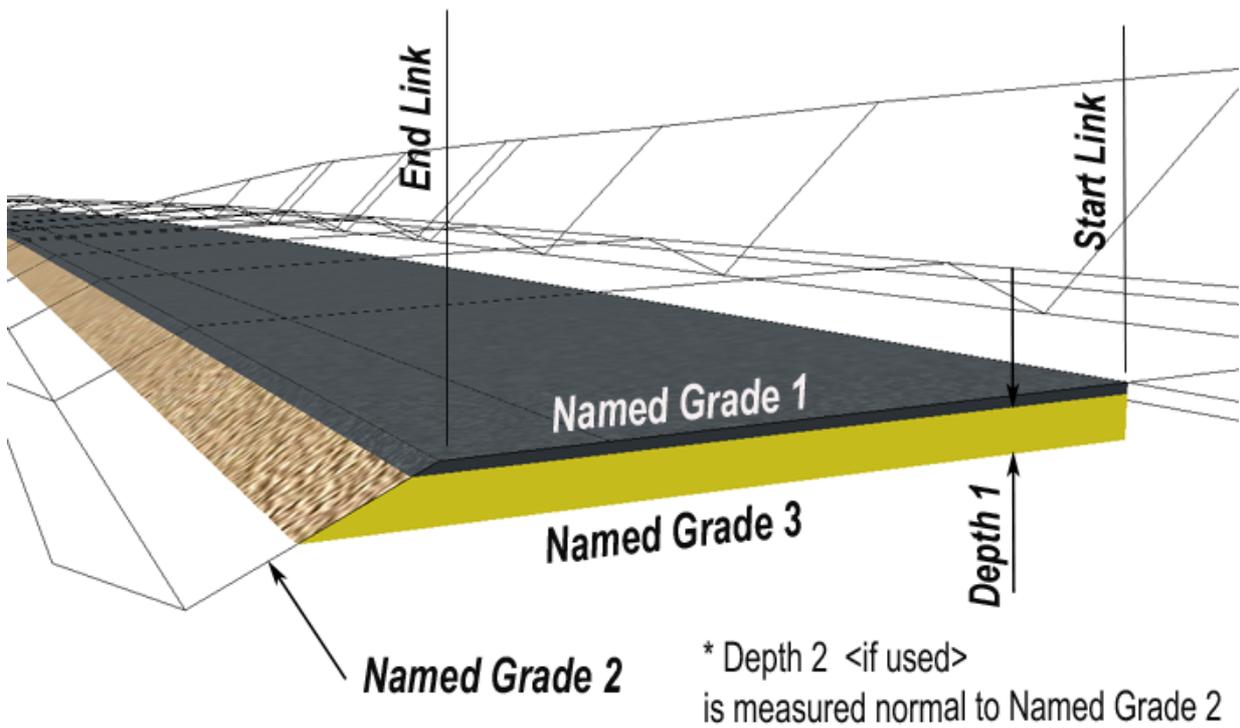
**Auto L/R**                      choice box                      No                      Yes / No

*If set to <No> then the Start and End Links remain unchanged.*

*If set to <Yes> then the Start and End Links need the suffix for L/R removed.*

*e.g Any <Design> string names like **ESL** would need the <L> removed*

*e.g Any <Layer> string names like **PVBALRO1** would need the <LRO1> removed*



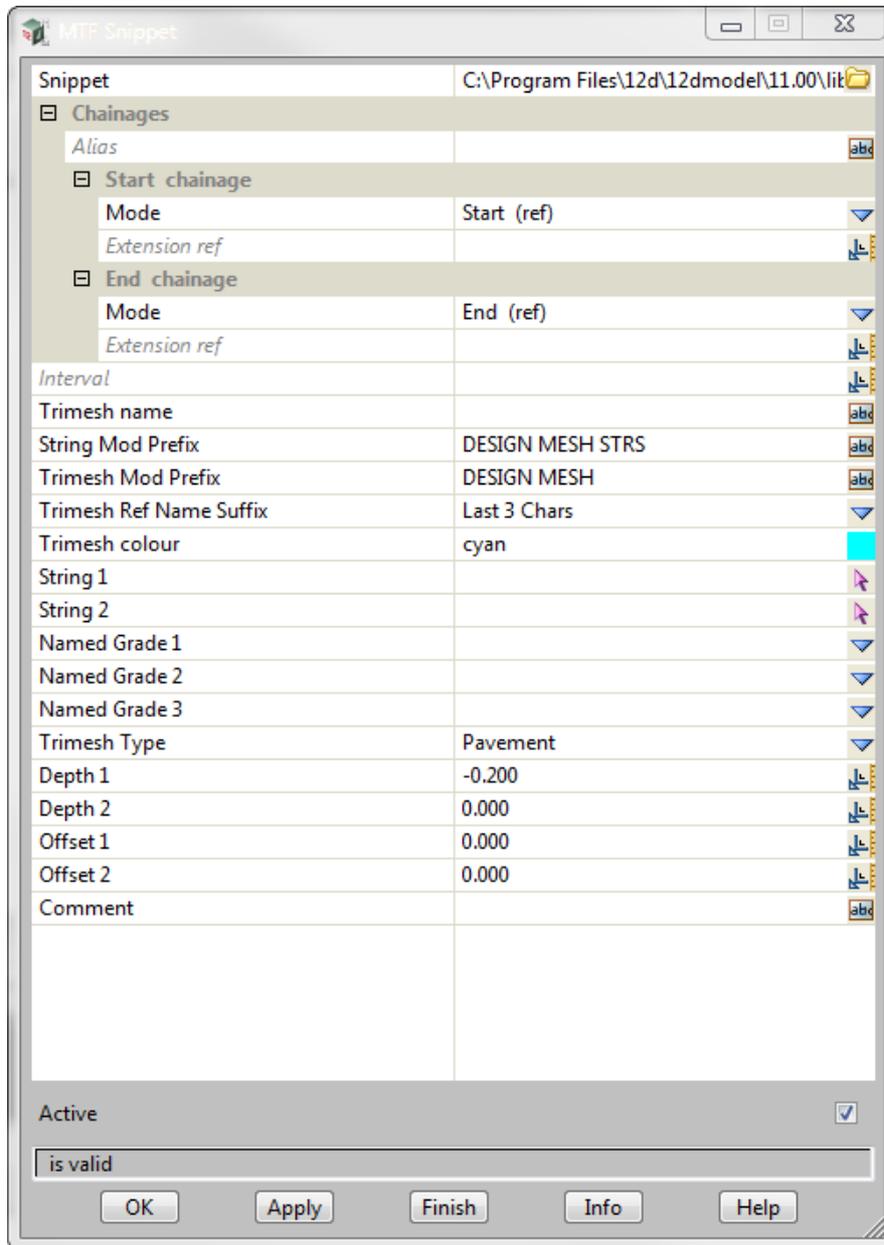
Continue to [22.3.1.5.6 TRI 2PT PAV NAMED GRADES EXT STRS](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

### 22.3.1.5.6 TRI\_2PT\_PAV\_NAMED\_GRADES\_EXT\_STRS

This Snippet is used to create typical pavement, using External strings NOT created in your MTF. It is linked to the String 1 and String 2 at a depth specified.

Offsets from these links can be set and measured Horizontally.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippet files

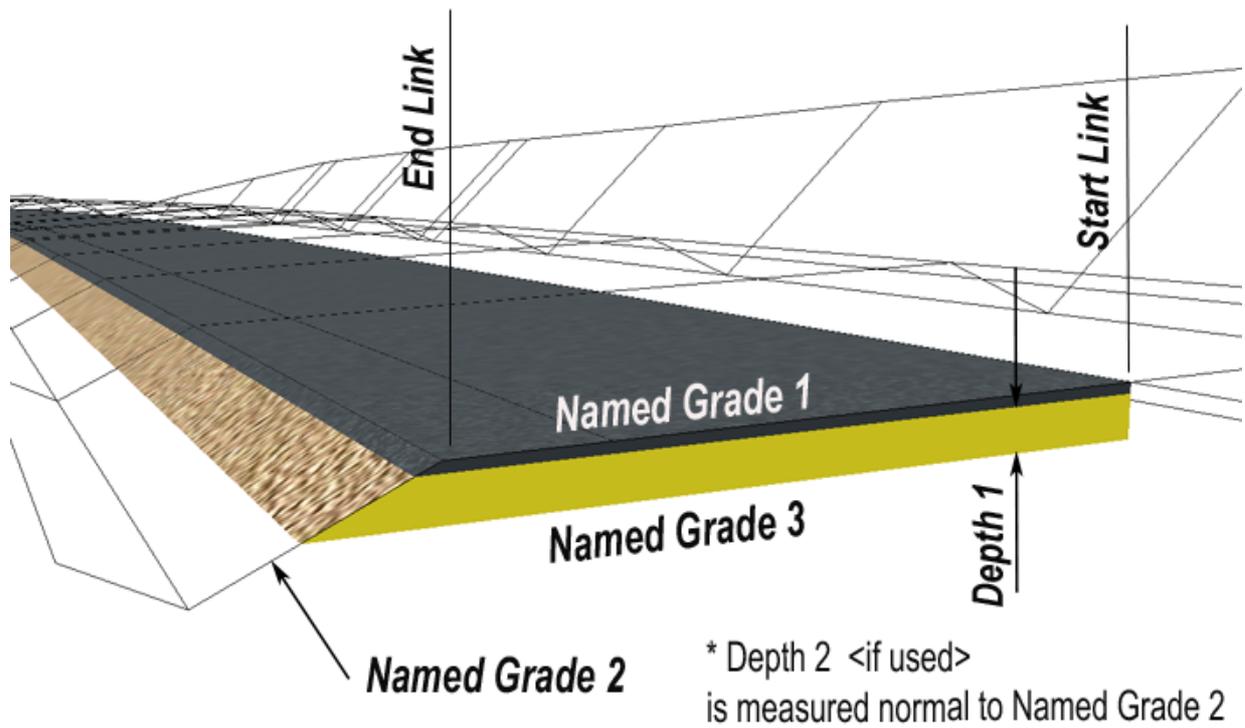
**Alias, Start Chainage, End Chainage, Interval**  
*defines the start and end chainages to apply a snippet.*

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

**Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>String 1</b>	select		
<b>String 2</b>	select		
<b>Named Grade 1</b>	choice box	Design	various
	<i>(To define the top surface)</i>		
<b>Named Grade 2</b>	choice box	Design	various
	<i>(To defined the typical batter slope)</i>		
<b>Named Grade 3</b>	choice box	Design	various
	<i>(To define the bottom surface)</i>		
<b>Trimesh Type</b>	choice box	Pavement	Pavement / Bottom Surface
<b>Depth 1</b>	real		-0.2
<b>Depth 2</b>	real		
	<i>(Optional...if used is measure Normal to Named Grade 2)</i>		
<b>Offset 1</b>	real		0
<b>Offset 2</b>	real		0



Continue to [22.3.1.5.7 TRI\\_2PT\\_PAV\\_STOP\\_NAMED\\_GRADES](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

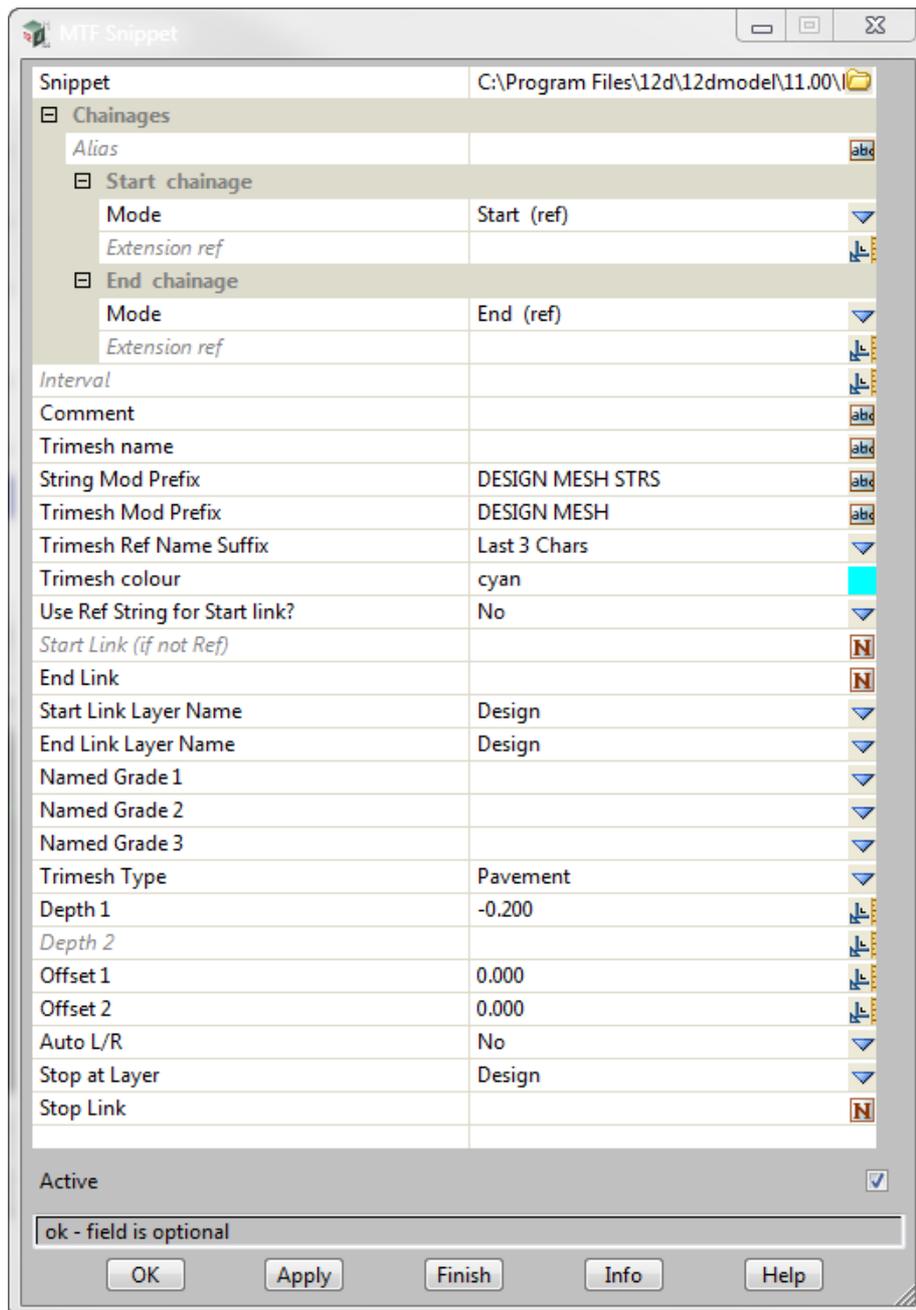
### 22.3.1.5.7 TRI\_2PT\_PAV\_STOP\_NAMED\_GRADES

This Snippet is used to create typical pavement, using <Design> or <Layer> strings created in your MTF.

It is linked to the Start and End Links at a depth specified.

Offsets from these links can be set and measured Horizontally or down the slope created from the two links.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

*snippet to run.*

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>Attach to Layer Name</b>	choice box	Design	various
<b>Use Ref String for Start Link?</b>	choice box	No	Yes / No
<b>Start Link (if not Ref)</b>	name		
	<i>(Optional if use ref string is set to Yes)</i>		
<b>End Link</b>	name		
<b>Start Link Layer Name</b>	choice box	Design	various
<b>End Link Layer Name</b>	choice box	Design	various
<b>Named Grade 1</b>	choice box	Design	various
	<i>(To define the top surface)</i>		
<b>Named Grade 2</b>	choice box	Design	various
	<i>(To defined the typical batter slope)</i>		
<b>Named Grade 3</b>	choice box	Design	various
	<i>(To define the bottom surface)</i>		
<b>Trimesh Type</b>	choice box	Pavement	Pavement / Bottom Surface
<b>Depth 1</b>	real		-0.2
<b>Depth 2</b>	real		
	<i>(Optional...if used is measure Normal to Named Grade 2)</i>		
<b>Offset 1</b>	real		0

<b>Offset 2</b>	real		0
<b>Auto L/R</b>	choice box	No	Yes / No

*If set to <No> then the Start and End Links remain unchanged.*

*If set to <Yes> then the Start and End Links need the suffix for L/R removed.*

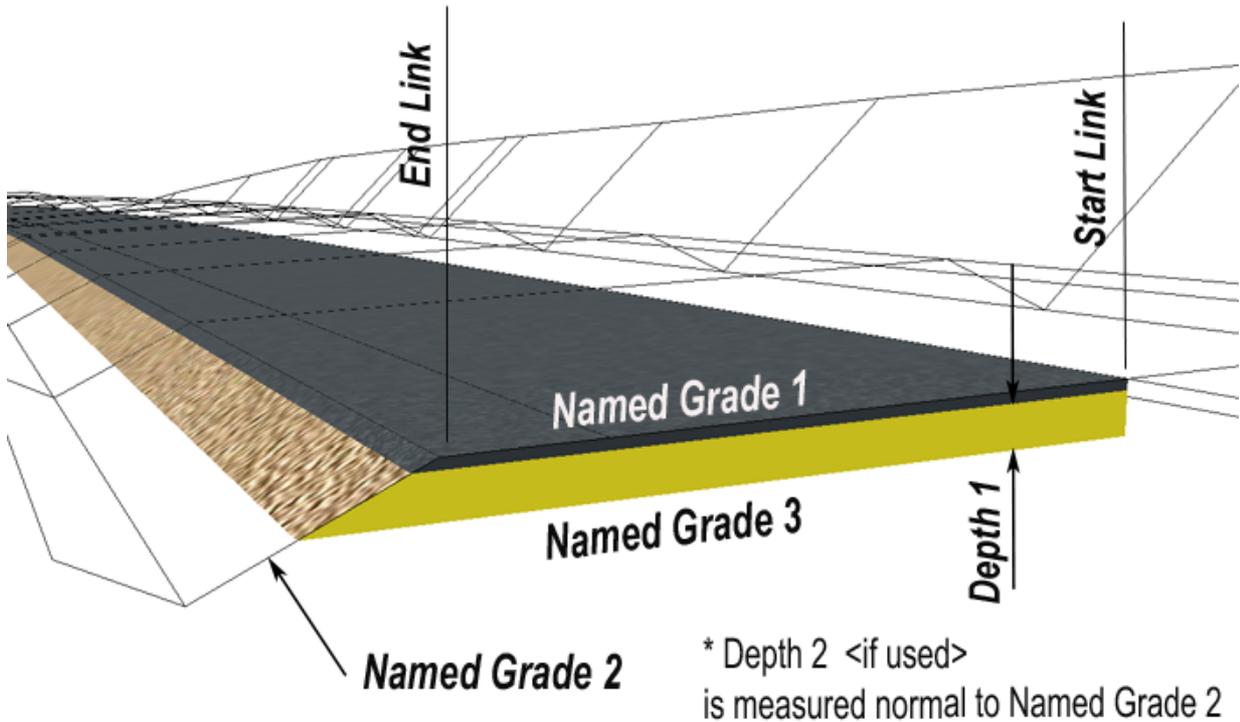
*e.g Any <Design> string names like **ESL** would need the <L> removed*

*e.g Any <Layer> string names like **PVBALRO1** would need the <LRO1> removed*

<b>Stop at Layer</b>	choice box	Design	various
----------------------	------------	--------	---------

**Stop Link** name

*(The stop layer and link are to be used to stop any pavement layer under a specific point such as a drain invert or interface point, rather than extending through at "Named Grade 2")*



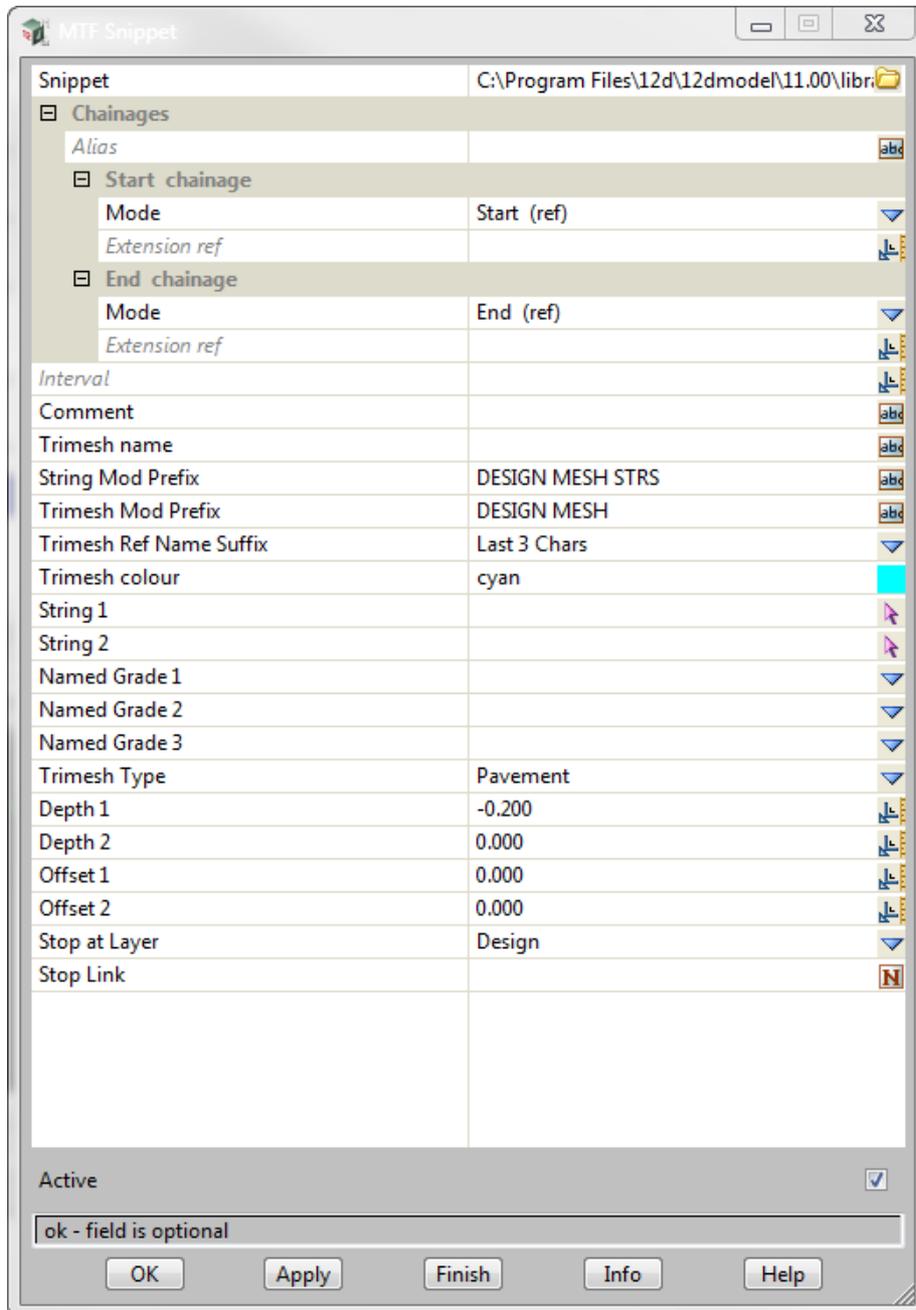
Continue to [22.3.1.5.8 TRI\\_2PT\\_PAV\\_STOP\\_NAMED\\_GRADES\\_EXT\\_STRS](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#) .

### 22.3.1.5.8 TRI\_2PT\_PAV\_STOP\_NAMED\_GRADES\_EXT\_STRS

This Snippet is used to create typical pavement, using External strings NOT created in your MTF. It is linked to the String 1 and String 2 at a depth specified.

Offsets from these links can be set and measured Horizontally.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippet files

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

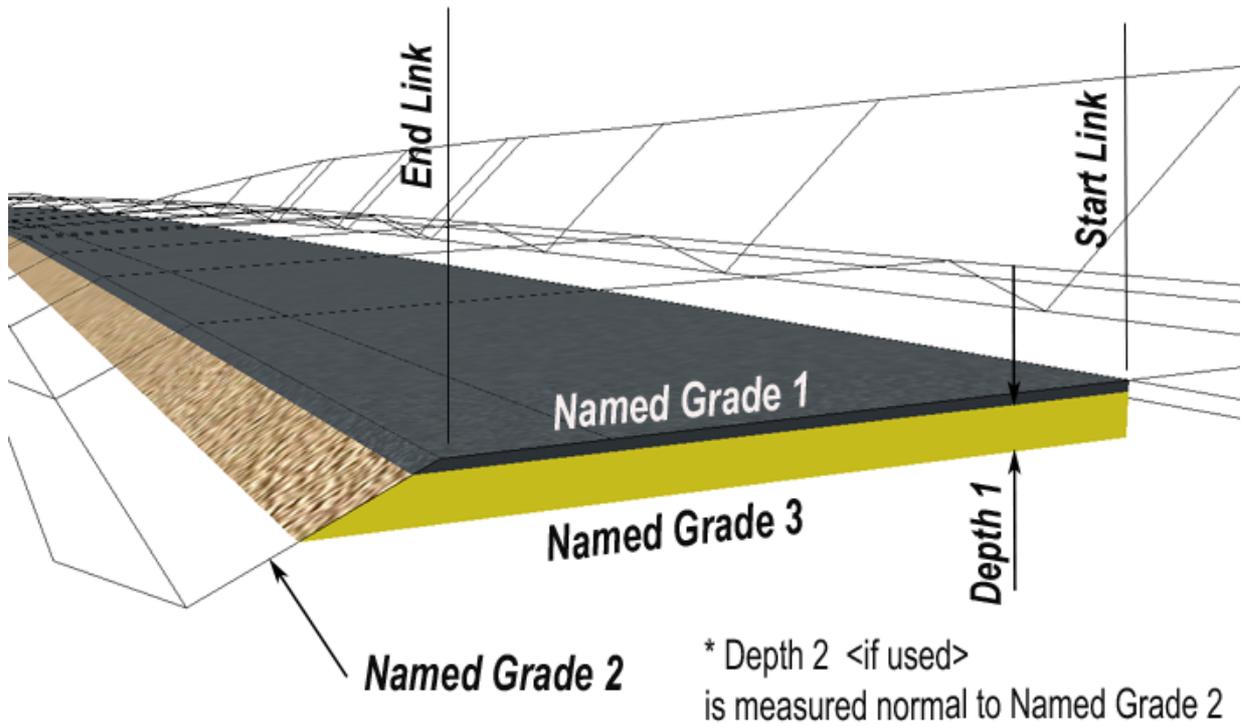
*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>String 1</b>	select		
<b>String 2</b>	select		
<b>Named Grade 1</b>	choice box	Design	various
	<i>(To define the top surface)</i>		
<b>Named Grade 2</b>	choice box	Design	various
	<i>(To defined the typical batter slope)</i>		
<b>Named Grade 3</b>	choice box	Design	various
	<i>(To define the bottom surface)</i>		
<b>Trimesh Type</b>	choice box	Pavement	Pavement / Bottom Surface
<b>Depth 1</b>	real		-0.2
<b>Depth 2</b>	real		
	<i>(Optional...if used is measure Normal to Named Grade 2)</i>		
<b>Offset 1</b>	real		0
<b>Offset 2</b>	real		0
<b>Stop at Layer</b>	choice box	Design	various
<b>Stop Link</b>	name		

*(The stop layer and link are to be used to stop any pavement layer under a specific point such as a drain invert or interface point, rather than extending through at "Named Grade 2")*



Continue to [22.3.1.5.9 TRI\\_2PT\\_STRUCT\\_FILL](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

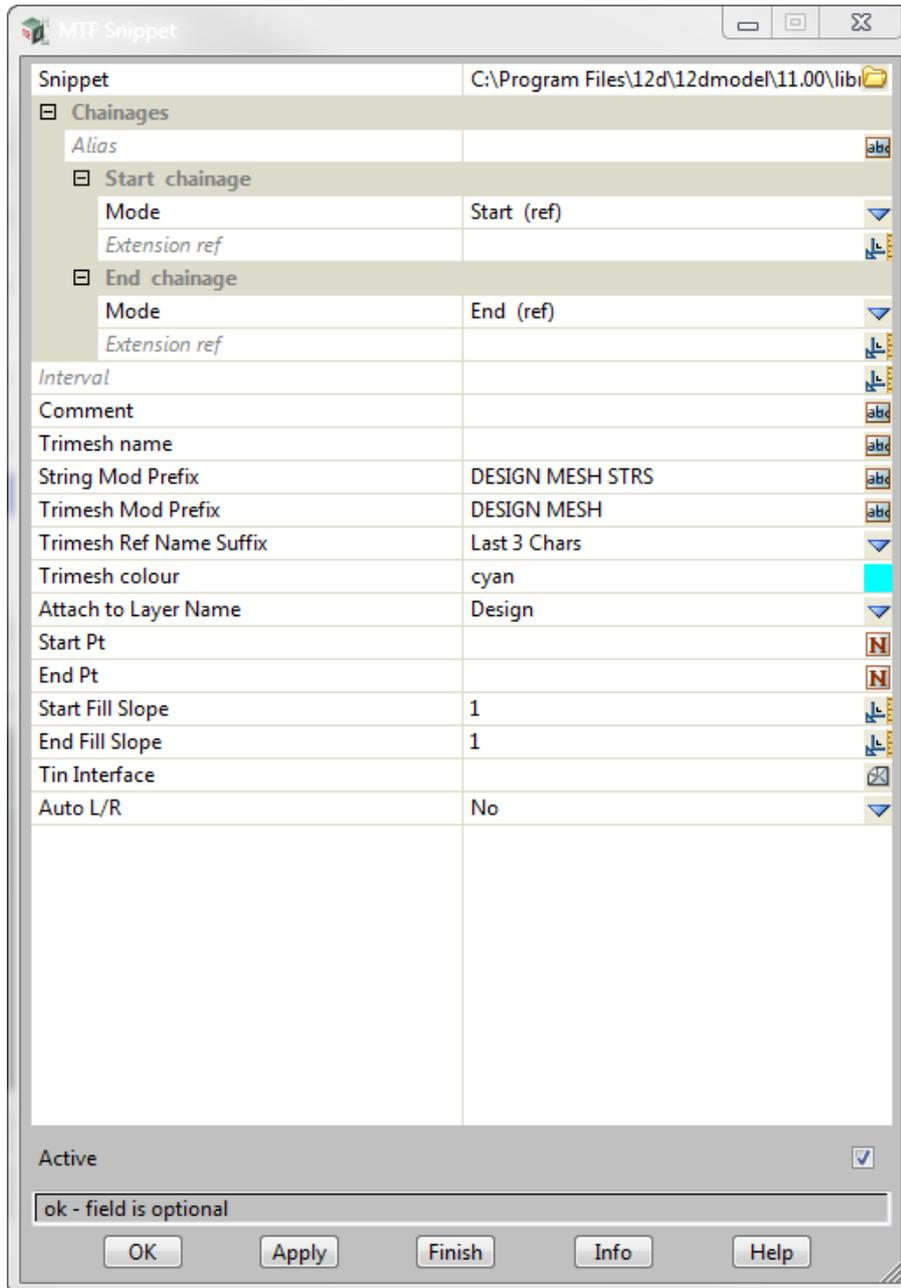
### 22.3.1.5.9 TRI\_2PT\_STRUCT\_FILL

This Snippet is used to create typical pavement, using <Design> or <Layer> strings created in your MTF.

It is linked to the Start and End Links at a slope specified.

Slopes are for fill, so no sign is required for either slope.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None

*(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)*

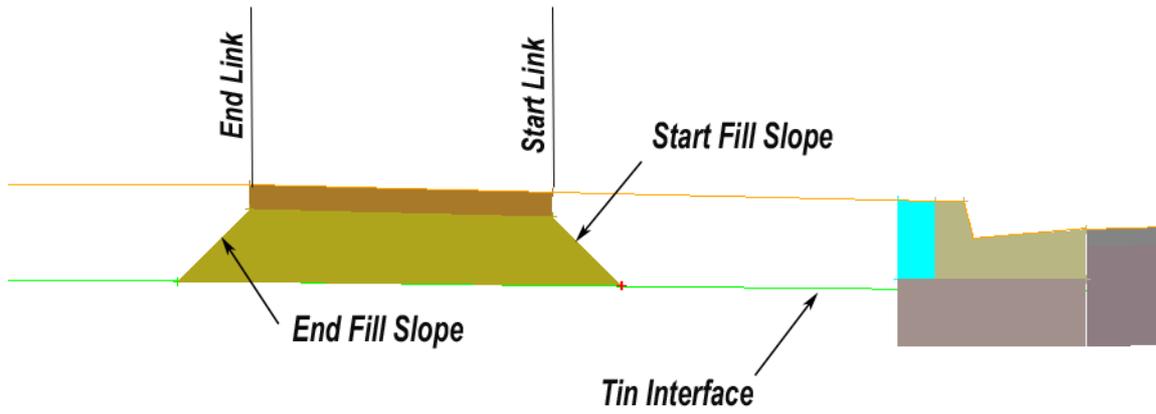
<b>Trimesh Colour</b>	colour box	cyan	various
<b>Attach to Layer Name</b>	choice box	Design	various
<b>Start Link</b>	name		
<b>End Link</b>	name		
<b>Start Fill Slope</b>	real		1
<b>End Fill Slope</b>	real		1
<b>Tin Interface</b>	tin		
<b>Auto L/R</b>	choice box	No	Yes / No

*if set to <No> then the Start and End Links remain unchanged.*

*If set to <Yes> then the Start and End Links need the suffix for L/R removed.*

*e.g Any <Design> string names like **ESL** would need the <L> removed*

*e.g Any <Layer> string names like **PVBALROI** would need the <LROI> removed*



Continue to [22.3.1.5.10 TRI\\_3PT\\_PAV\\_DES\\_LAY](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).



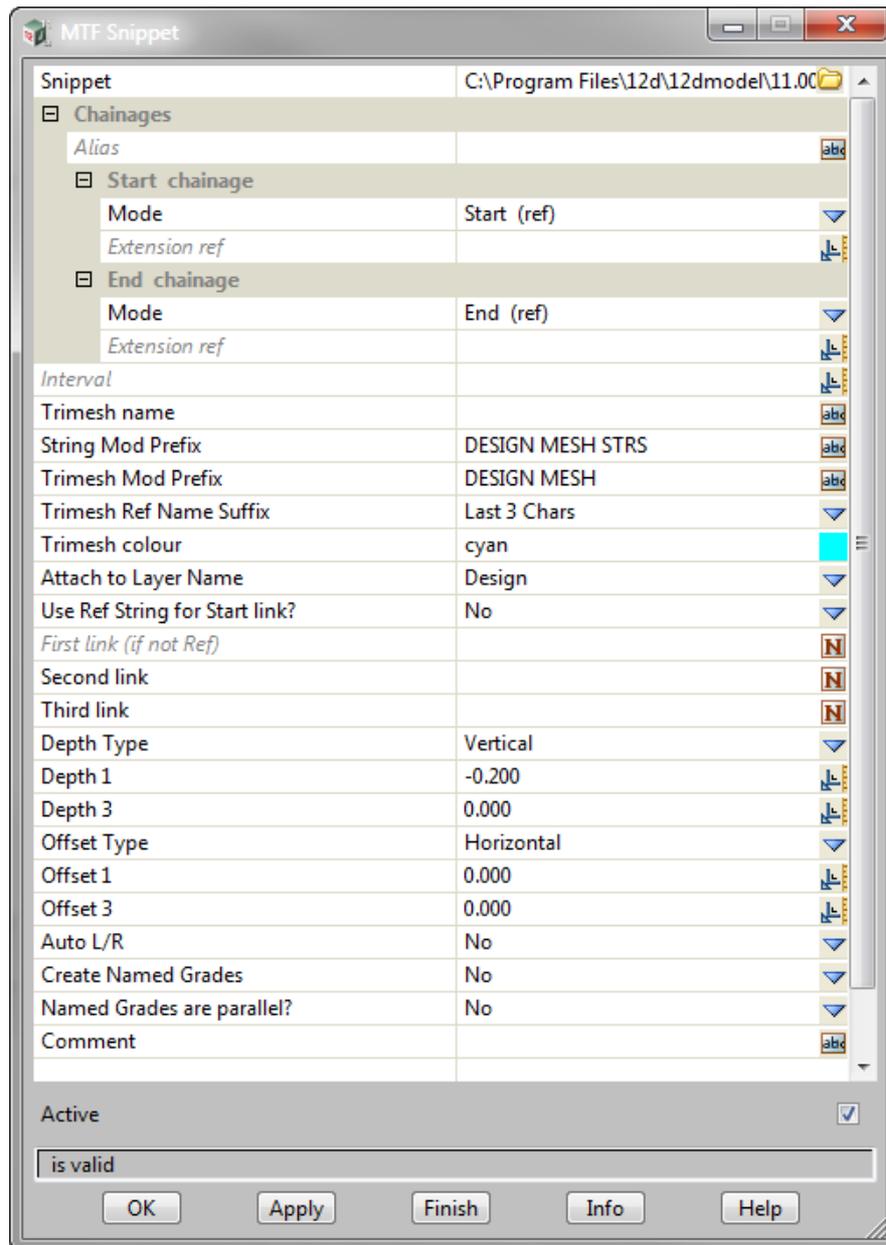
### 22.3.1.5.10 TRI\_3PT\_PAV\_DES\_LAY

This Snippet is used to create typical pavement, using <Design> or <Layer> strings created in your MTF.

It is linked to the First, Second and Third Links at depths specified.

Offsets from these links can be set and measured Horizontally or down the slope created from the links.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

**Alias, Start Chainage, End Chainage, Interval**

defines the start and end chainages to apply a snippet.

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

**Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>Attach to Layer Name</b>	choice box	Design	various
<b>Use Ref String for Start Link?</b>	choice box	No	Yes / No
<b>First Link (if not Ref)</b>	name		
	<i>(Optional if use ref string is set to Yes)</i>		
<b>Second Link</b>	name		
<b>Third Link</b>	name		
<b>Depth Type</b>	choice box	Vertical	Vertical Normal Top Surface Bottom Surface
<b>Depth 1</b>	real		-0.2
	<i>(Measured from First Link)</i>		
<b>Depth 3</b>	real		-0.2
	<i>(Measured from Third Link)</i>		
<b>Offset Type</b>	choice box	Horizontal	Horizontal On Slope
<b>Offset 1</b>	real		0
	<i>(Measured from First Link)</i>		
<b>Offset 3</b>	real		0
	<i>(Measured from Third Link)</i>		

**Auto L/R** choice box No Yes / No

If set to <No> then the Start and End Links remain unchanged.

If set to <Yes> then the Start and End Links need the suffix for L/R removed.

e.g Any <Design> string names like **ESL** would need the <L> removed

e.g Any <Layer> string names like **PVBALRO1** would need the <LRO1> removed

**Create Named Grades** choice box No Yes / No

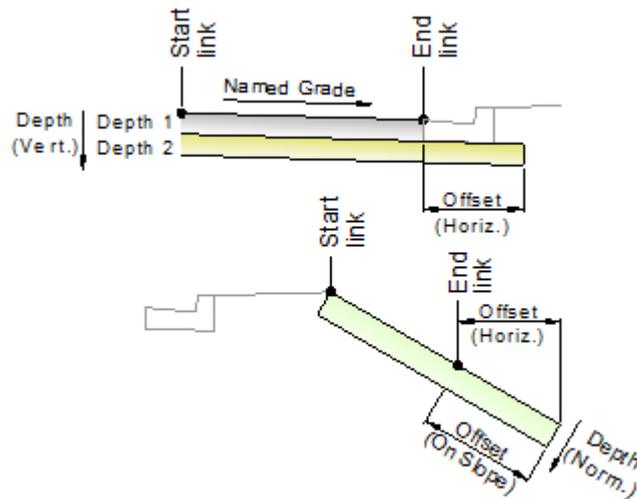
if set to <Yes> then a Named Grade in the format <Start Link Name \_ End Link Name> is created for use in the MTF by other modifiers

**Named Grades are Parallel?** choice box No Yes / No

(This is a decision by the user in case the three links form one grade.

If **No** then the middle point is calculated as an intersection of two grades, using "Depth 1" & "Depth 3".

If **Yes** then the middle point is calculated from the Second Link at the "Depth 1")



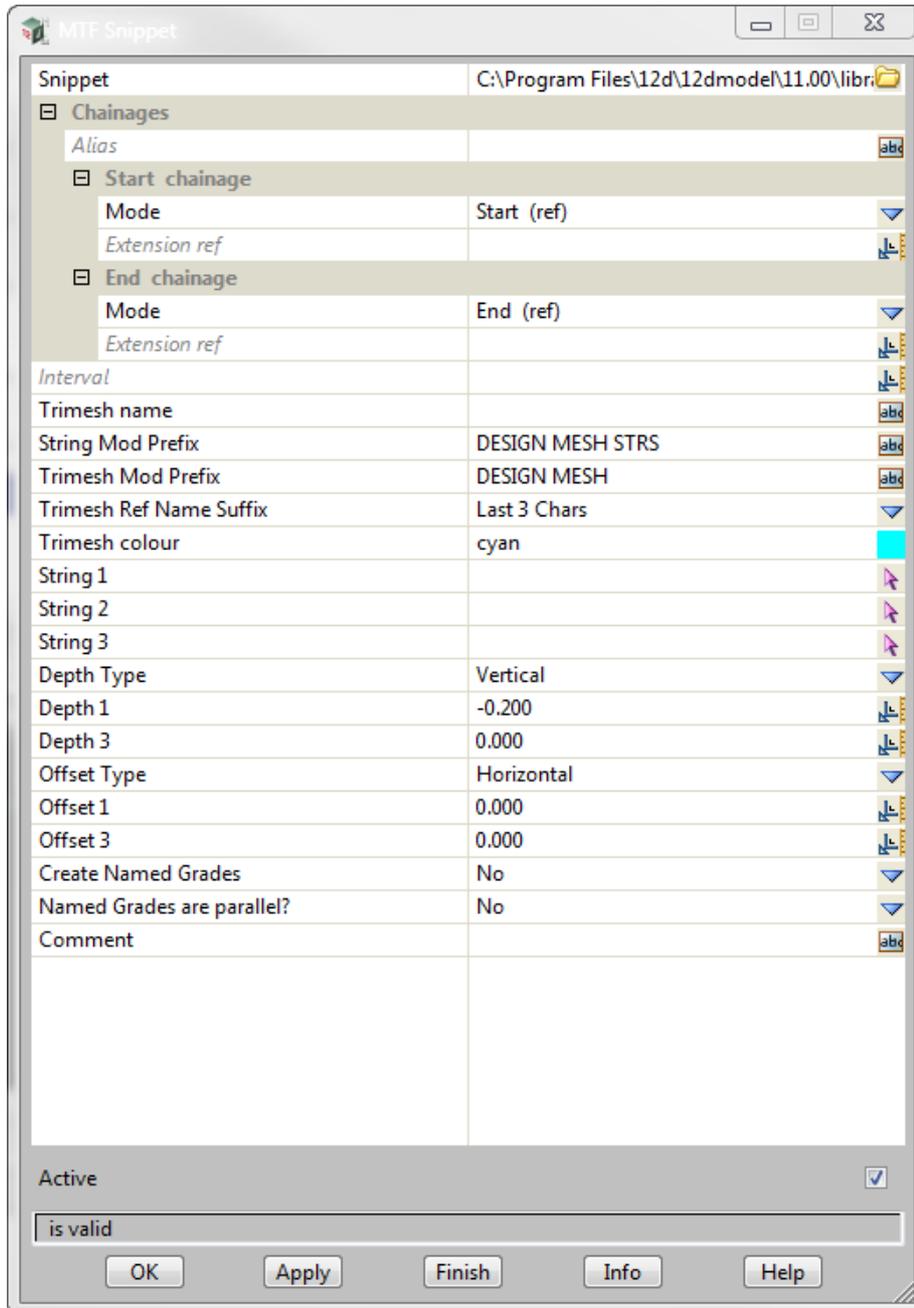
Continue to [22.3.1.5.11 TRI\\_3PT\\_PAV\\_EXT\\_STRS](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

### 22.3.1.5.11 TRI\_3PT\_PAV\_EXT\_STRS

This Snippet is used to create typical pavement, using External strings NOT created in your MTF. It is linked to the three strings at depths specified.

Offsets from these links can be set and measured Horizontally or down the slope created from the links.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

**Alias, Start Chainage, End Chainage, Interval**

*defines the start and end chainages to apply a snippet.*

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

**Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.*

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None
	<i>(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)</i>		
<b>Trimesh Colour</b>	colour box	cyan	various
<b>Attach to Layer Name</b>	choice box	Design	various
<b>Use Ref String for Start Link?</b>	choice box	No	Yes / No
<b>String 1</b>	select		
<b>String 2</b>	select		
<b>String 3</b>	select		
<b>Depth Type</b>	choice box	Vertical	Vertical Normal Top Surface Bottom Surface
<b>Depth 1</b>	real		-0.2
	<i>(Measured from First Link)</i>		
<b>Depth 3</b>	real		-0.2
	<i>(Measured from Third Link)</i>		
<b>Offset Type</b>	choice box	Horizontal	Horizontal On Slope
<b>Offset 1</b>	real		0
	<i>(Measured from First Link)</i>		
<b>Offset 3</b>	real		0
	<i>(Measured from Third Link)</i>		

**Create Named Grades** choice box No Yes / No

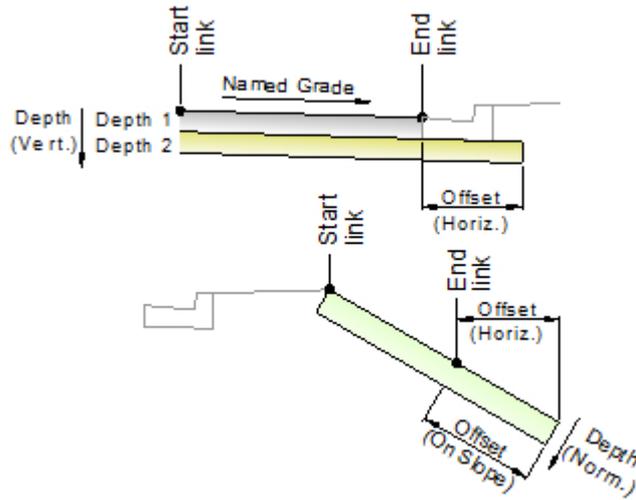
if set to <Yes> then a Named Grade in the format <Start Link Name \_ End Link Name> is created for use in the MTF by other modifiers

**Named Grades are Parallel?** choice box No Yes / No

(This is a decision by the user in case the three links form one grade.

If **No** then the middle point is calculated as an intersection of two grades, using "Depth 1" & "Depth 3".

If **Yes** then the middle point is calculated from the Second Link at the "Depth 1")



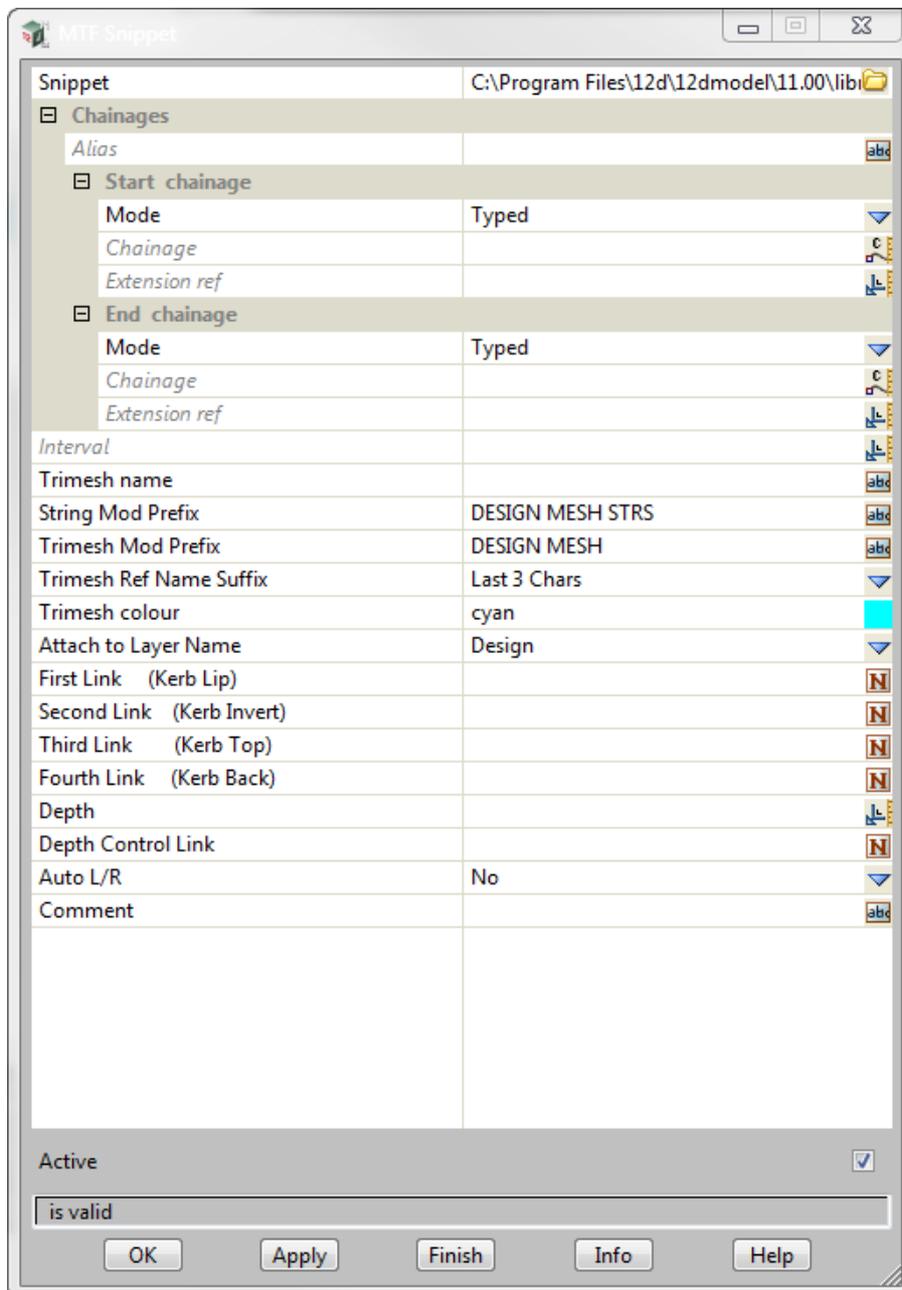
Continue to [22.3.1.5.12 TRI\\_KERB\\_PROFILE](#) or return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#).

### 22.3.1.5.12 TRI\_KERB\_PROFILE

This Snippet is used to create typical kerb profile, using <Design> or <Layer> strings created in your MTF.

It is defined by four Links at a depth specified by one of those links.

The Reference string name can be used as part of the model naming convention.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b> <i>snippet to run.</i>	snippet box	*.mtfsnippet,	*.mtfsnippetc files

**Alias, Start Chainage, End Chainage, Interval**

defines the start and end chainages to apply a snippet.

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

**Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see 17.2.2.1.1 Common Fields and Buttons on MTF Modifier Panels.

<b>Trimesh Name</b>	input		
	<i>Trimesh name that is used in model names below</i>		
<b>String Mod Prefix</b>	input		DESIGN MESH STRS
<b>Trimesh Mod Prefix</b>	input		DESIGN MESH
<b>Trimesh Ref Name Suffix</b>	choice box	Last 3 Chars	All Chars Last Char Last 2 Chars Last 3 Chars Last 4 Chars None

*(Model syntax: String Mod Prefix Trimesh Name Trimesh Ref Name Suffix)*

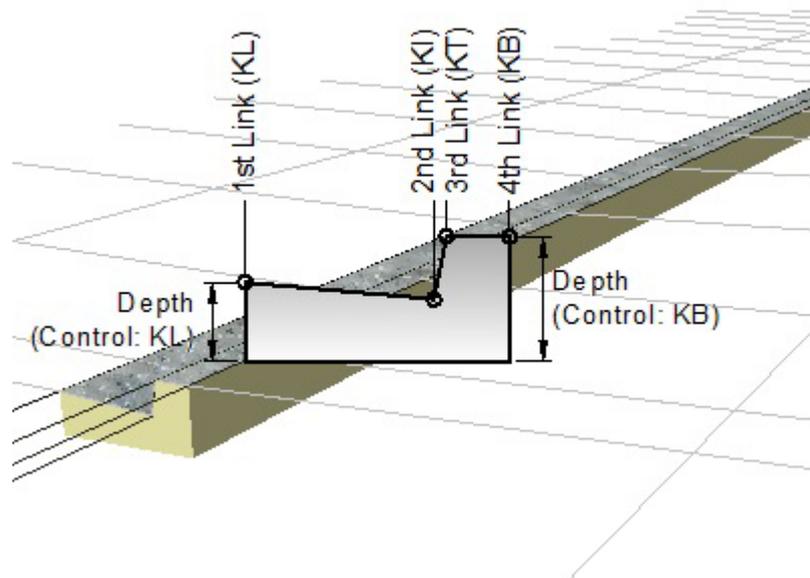
<b>Trimesh Colour</b>	colour box	cyan	various
<b>Attach to Layer Name</b>	choice box	Design	various
<b>First Link</b>	name		
<b>Second Link</b>	name		
<b>Third Link</b>	name		
<b>Fourth Link</b>	name		
<b>Depth</b>	real		
<b>Depth Control Link</b>	name		
<b>Auto L/R</b>	choice box	No	Yes / No

*if set to <No> then the Start and End Links remain unchanged.*

*If set to <Yes> then the Start and End Links need the suffix for L/R removed.*

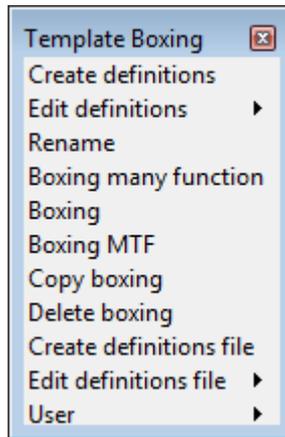
*e.g Any <Design> string names like **ESL** would need the <L> removed*

*e.g Any <Layer> string names like **PVBALROI** would need the <LROI> removed*



Return to [22.3.1.5 12d Supplied Snippets](#) or [22.3.1 Snippets](#) .

## 22.4 Boxing



There are new options to **Rename**, **Copy** and **Delete** the **bf** file.

The **Create definitions file** and **Edit definitions file** options have been moved to just above **User** on the **Template Boxing** menu.

Also see

[22.4.1 Create/Edit Definitions](#)

[22.4.2 Rename Boxing File](#)

[22.4.3 Copy Boxing](#)

[22.4.4 Delete Boxing](#)

[22.4.5 Smart Chainages in Boxing Many Function](#)

### 22.4.1 Create/Edit Definitions

See

[22.4.1.1 Boxing Rules on Edit Boxing Definitions Now a Grid](#)

[22.4.1.2 Active Tick Box Column](#)

[22.4.1.3 Regions in the Boxing Rules Grid](#)

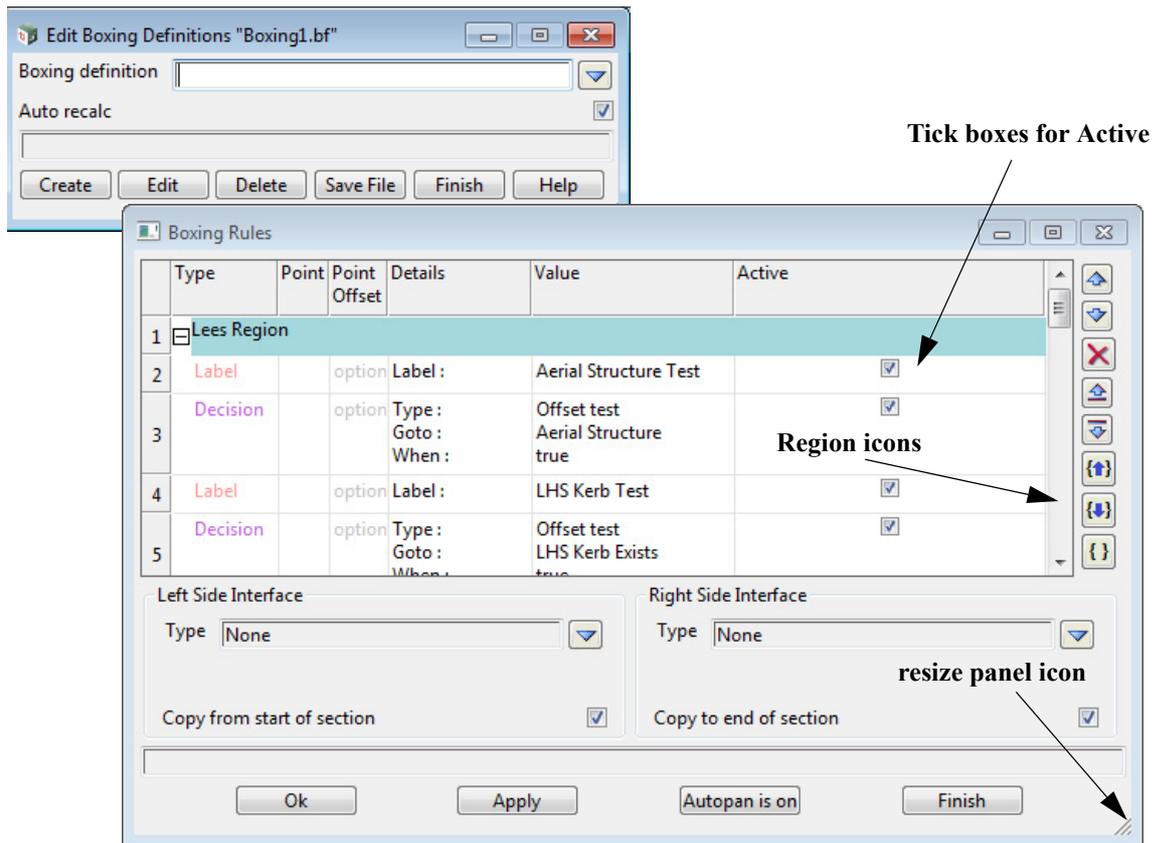
#### 22.4.1.1 Boxing Rules on Edit Boxing Definitions Now a Grid

When *Creating* or *Editing* a **Boxing Definitions** with the option

**Design =>Boxing =>Create definitions**

**Design =>Boxing =>Edit definitions**

the **Create** and **Edit** buttons on the **Edit Boxing Definitions** panel brings up a new grid.



As before, there are buttons on the right hand side to move rows **Up** and **Down**, **delete** a row, **insert** a blank row **Above/Below** the highlighted row.

The panel can be **resized** using the Resize panel on the bottom left right hand corner of the panel.

### 22.4.1.2 Active Tick Box Column

The **Active** column now has tick boxes instead of Yes/No choice boxes.

If the **Active** column for a command is ticked **on** then the command is used.

If the **Active** column for a command is not ticked, the command is not used.

### 22.4.1.3 Regions in the Boxing Rules Grid

There is a new **Region** command (rule) for the **Boxing Rules** grid.

A **Region** in a grid is a special command that has the property that a Region can be **collapsed**. And **collapsing** a Region means that all the rows after the Region command until the next Region command, are hidden in the grid. The row of the **Region** command, are hidden in the grid. The row of the **Region** command is coloured light blue.

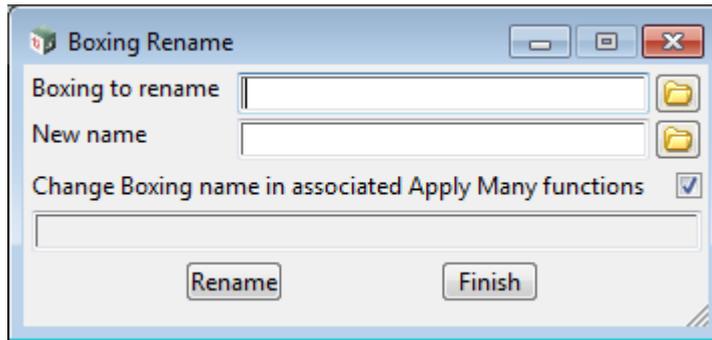
Regions must have a unique name within the Boxing Rules grid.

See [12.11 Regions in Some Grids](#)

## 22.4.2 Rename Boxing File

**Position of option on menu:** Design =>Boxing =>Rename

Selecting **Rename** brings up the **Boxing Rename** panel.



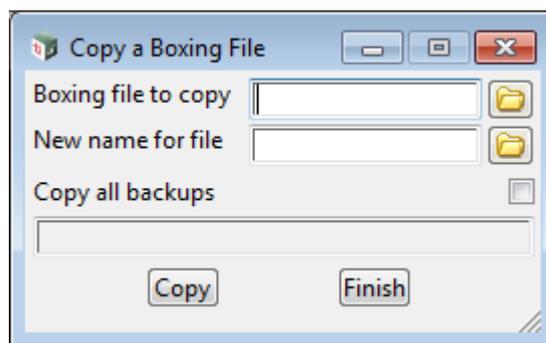
The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Boxing to rename</b> <i>name of the boxing bf file to rename</i>	file box		*.bf files
<b>New name</b> <i>new name for the boxing bf file</i>	file box		*.bf files
<b>Change Boxing name is associated Apply MTF functions</b> <i>if ticked, rename the bf file in all functions that include it.</i>	tick box		
<b>Rename</b> <i>rename the boxing bf file to the new name</i>	button		

### 22.4.3 Copy Boxing

Position of option on menu: **Design =>Boxing =>Copy**

Selecting **Copy** brings up the **Copy a Boxing File** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Boxing file to copy</b> <i>name of the boxing bf file to copy</i>	file box		*.bf files
<b>New name for file</b>	file box		*.bf files

*new name for the boxing bf file*

**Copy all backups**            tick box

*if ticked, copy all the backup files of the boxing bf file as well*

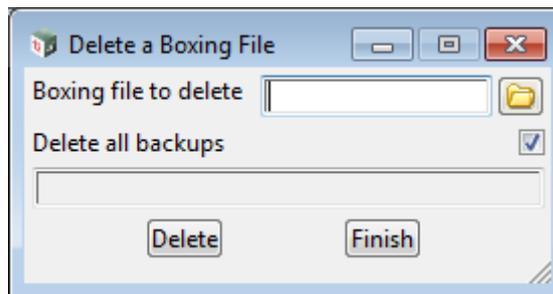
**Copy**                            button

*copy the boxing bf file to the new name*

## 22.4.4 Delete Boxing

**Position of option on menu:** Design =>Boxing =>Delete boxing

Selecting Delete brings up the **Delete a Boxing File** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Boxing file to delete</b> <i>name of the boxing bf file to delete</i>	file box		*.bf files
<b>Delete all backups</b> <i>if ticked, delete all the backup files of the boxing bf as well</i>	tick box		
<b>Delete</b> <i>delete the boxing bf file</i>	button		

## 22.4.5 Smart Chainages in Boxing Many Function

See

[22.4.5.1 Smart Chainages on Boxing Many Front Panel](#)

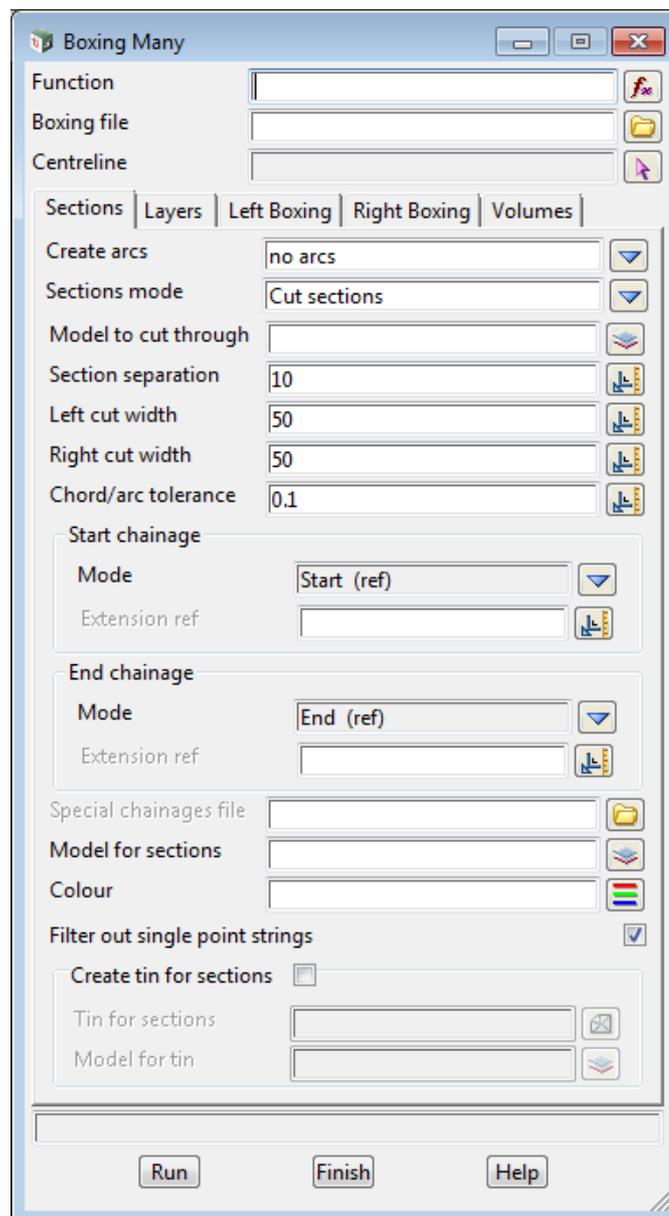
[22.4.5.2 Smart Chainages on Left/Right Boxing Tabs](#)

### 22.4.5.1 Smart Chainages on Boxing Many Front Panel

The **Boxing Many** Function now has **Smart Chainages** for its **Start Chainage** and **End Chainage**.

And yes, it is now a monster of a panel and won't fit on small screens

For the new information on **Smart Chainage**, see [9. Smart Chainages](#).



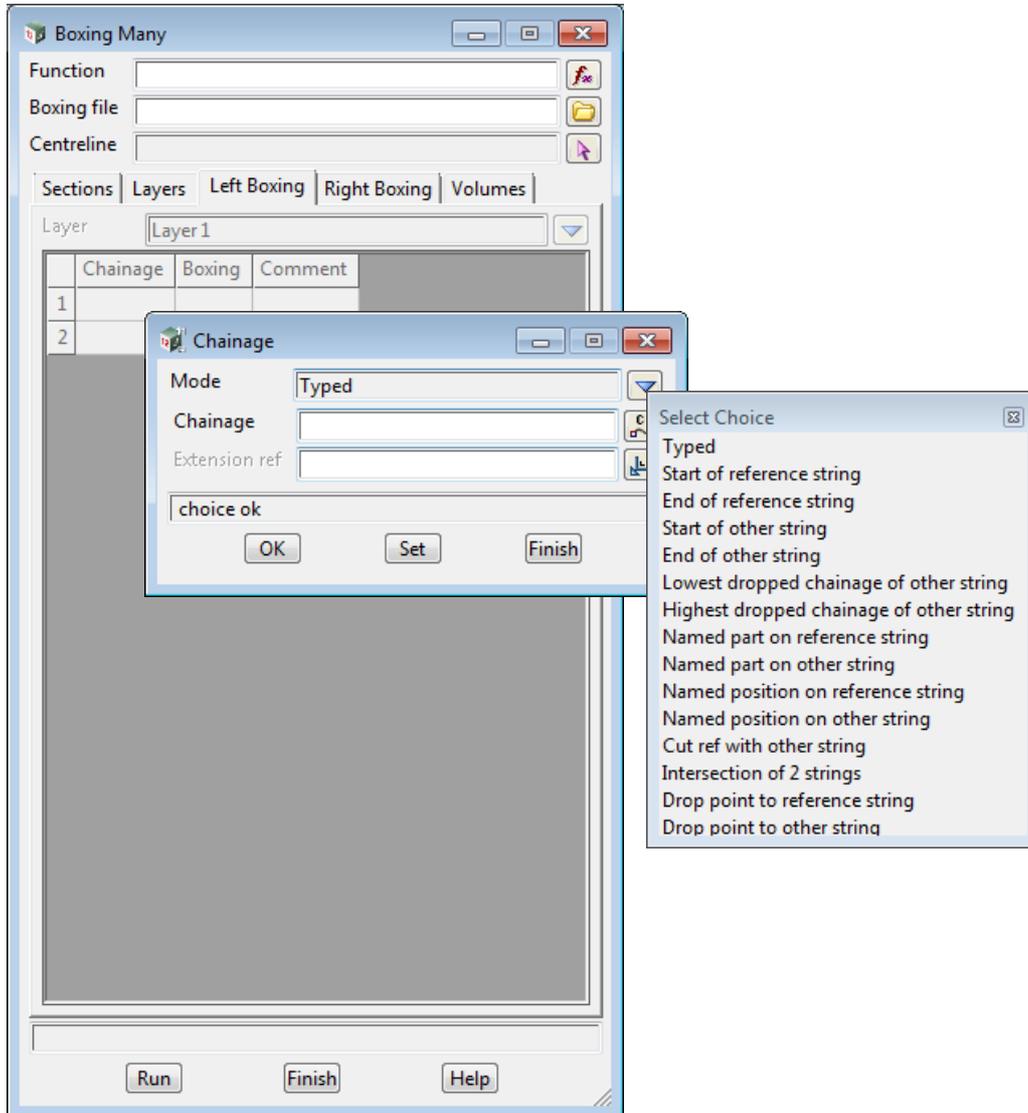
### 22.4.5.2 Smart Chainages on Left/Right Boxing Tabs

**Smart Chainages** are now being used in **Left** and **Right** tabs of the **Boxing Many Function**

**Design =>Boxing =>Boxing many function**

If you right click in the **Chainage** column in the **Left Boxing** and **Right Boxing** tabs, or select **Browse** if the Browse menu comes up) then a **Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**.

For the new information on **Smart Chainage**, see [9. Smart Chainages](#).



## 22.5 Roads

See

[22.5.1 Components](#)

[22.5.2 Create Roads - Enhanced](#)

### 22.5.1 Components

See

[22.5.1.1 RIGHT TURN MEDIAN](#)

[22.5.1.2 MAJOR MINOR ROAD INTERSECTION](#)

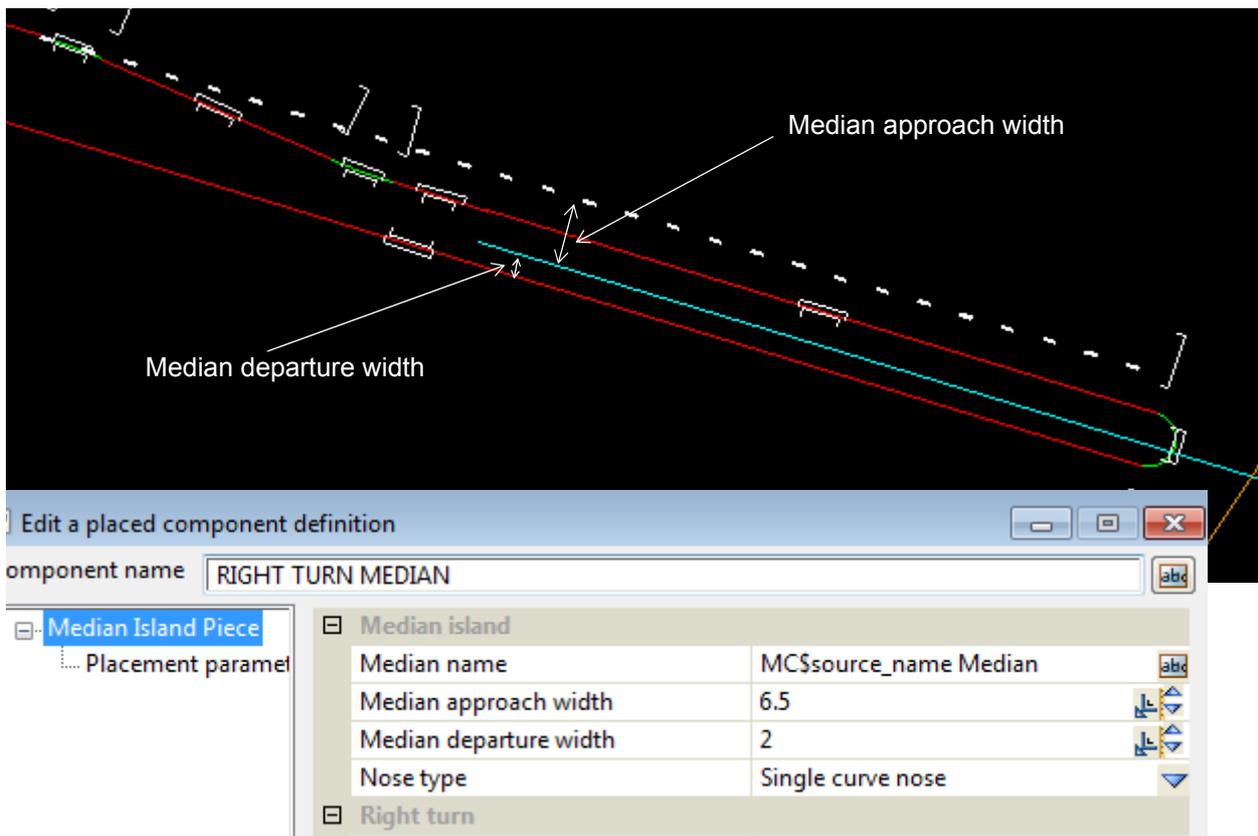
[22.5.1.3 HAMMERHEAD](#)

#### 22.5.1.1 RIGHT TURN MEDIAN

For the Components RIGHT TURN MEDIAN:

**Median width** is now called **Median approach width**.

There is a new **Median departure width** which allows the median to straddle the centre line.



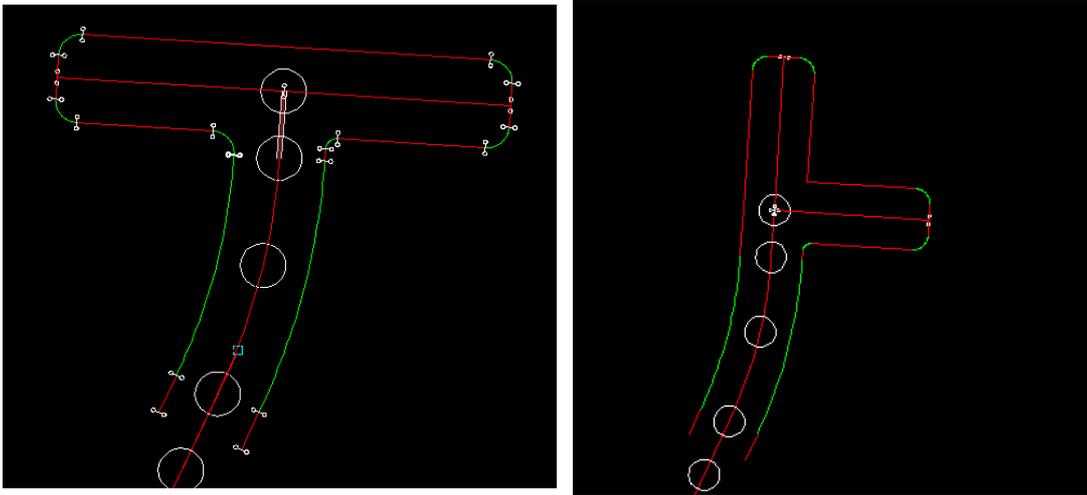
#### 22.5.1.2 MAJOR MINOR ROAD INTERSECTION

For the Components MAJOR MINOR ROAD INTERECTION:

Something similar to RIGHT TURN MEDIAN happens with **Median width** so that the Median can straddle the centre line but I'm unfamiliar with this Component and am having trouble explaining it.

### 22.5.1.3 HAMMERHEAD

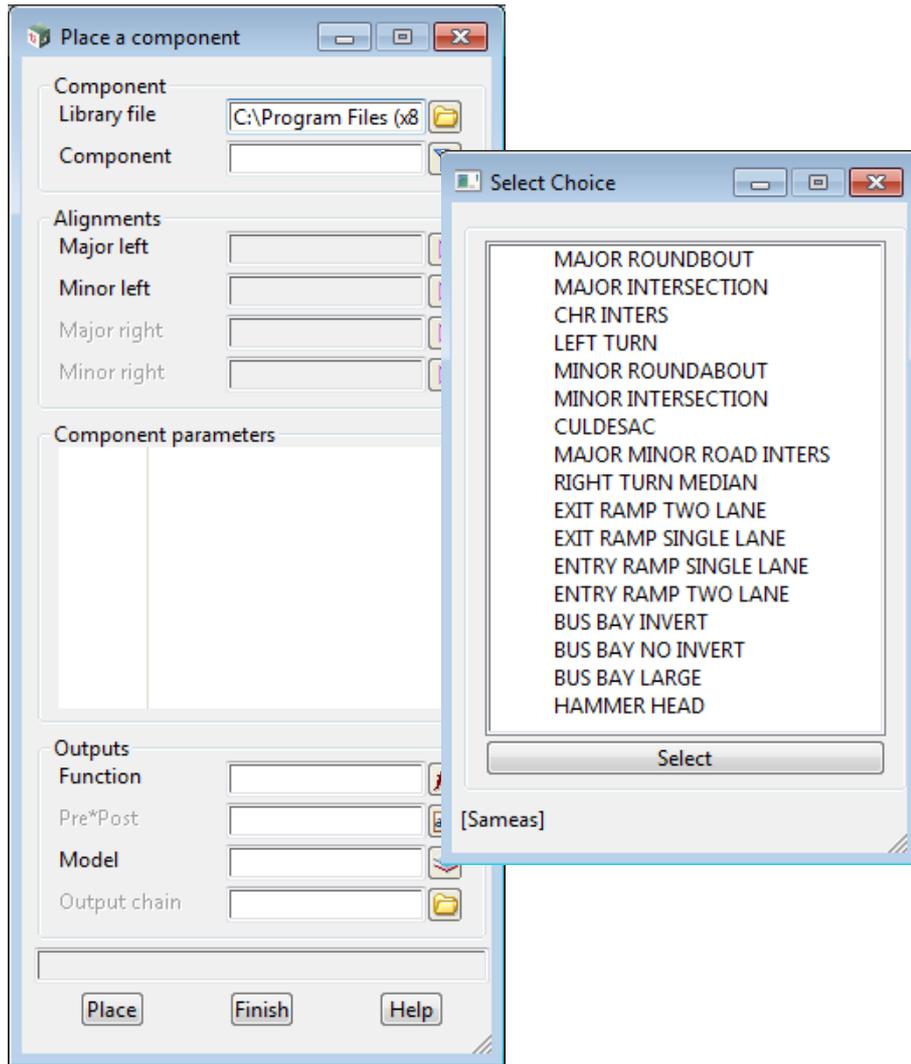
There is a new Component called a HAMMERHEAD with parameters to cover most situations. For example



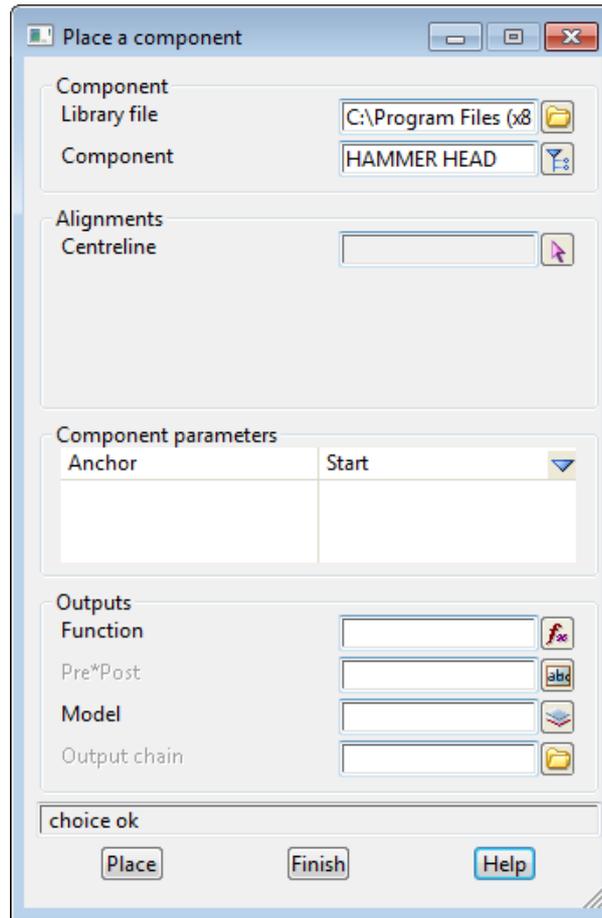
Selecting

**Design =>Roads =>Components =>Place component**

brings up the **Place a Component** panel.



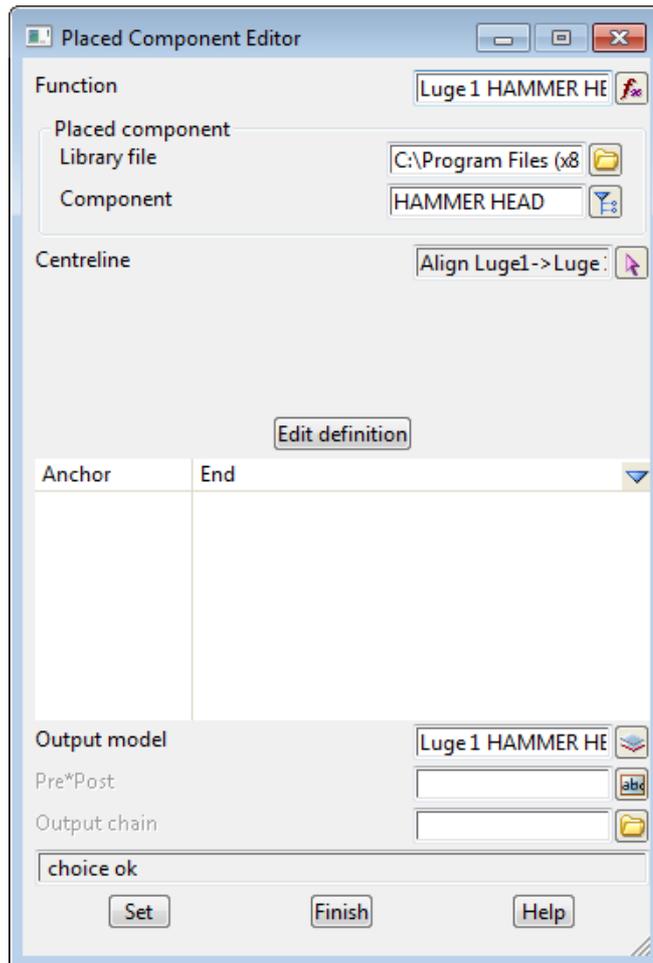
Clicking on the **Component** choice icon and selecting HAMMERHEAD puts information specific to a HAMMERHEAD on the panel.



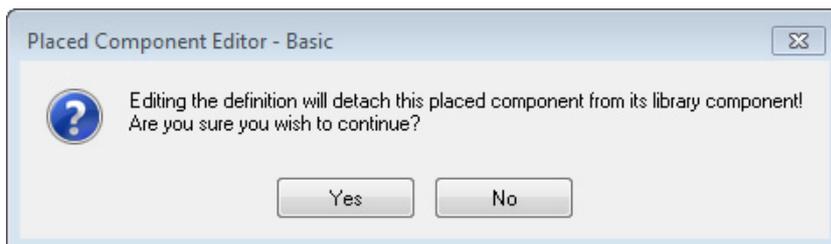
If you select a **Centreline**, the **Function** and **Model** boxes are automatically filled out using a combination of the string name, the word HAMMERHEAD and a unique number amongst the HAMMERHEAD functions already in the project.

**Anchor** says whether the HAMMERHEAD is to go at the **Start** or **End** of the Centreline.

Click **Place** to create the Hammerhead component.



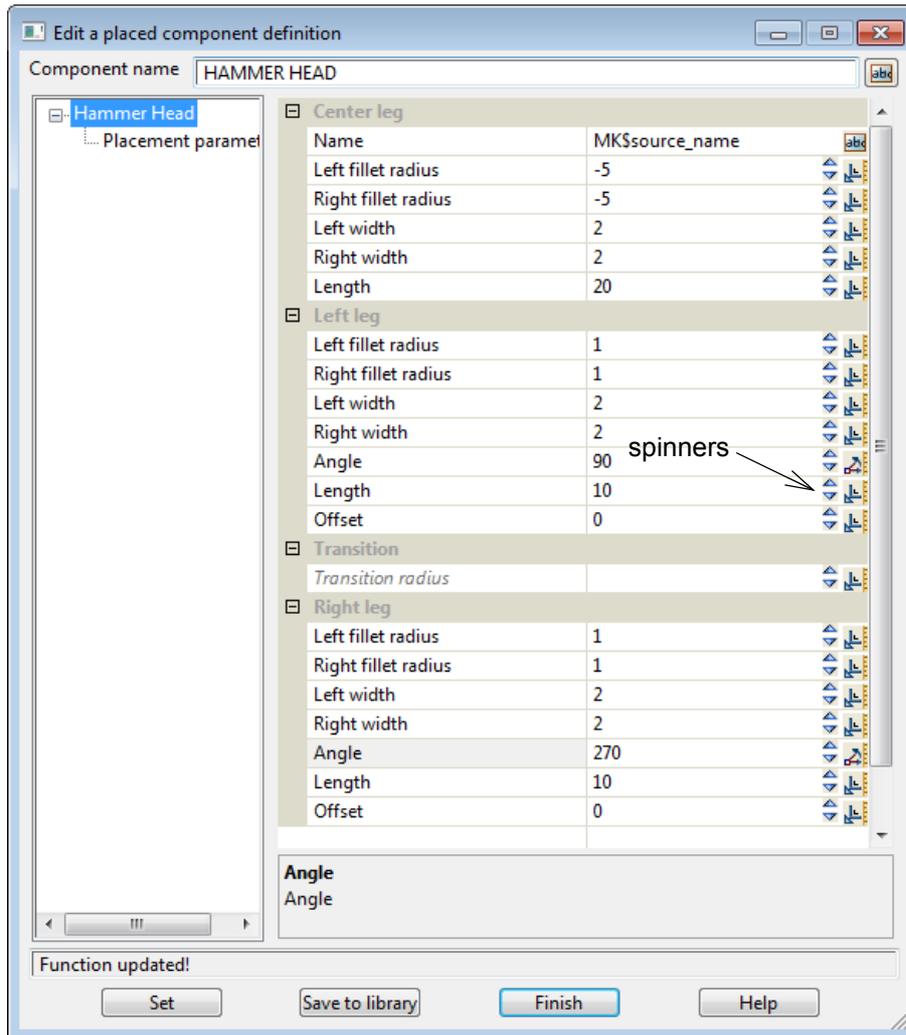
Clicking **Edit** definitions will first bring up a panel to let you know that editing the definition will modify the Hammerhead from the one that was in the library.



Clicking on **Yes** opens the **Edit a Placed Component Definition** panel.

Clicking on the tab **Placement parameters** will show the field **Anchor** (which was set before) but can be changed to either **Start** and **End**.

Clicking on **Hammer Head** displays the general tab of parameters.



The values in the fields can be modified and the Hammerhead will dynamically change. Clicking on the spinners will increase/decrease the value by a certain amount.

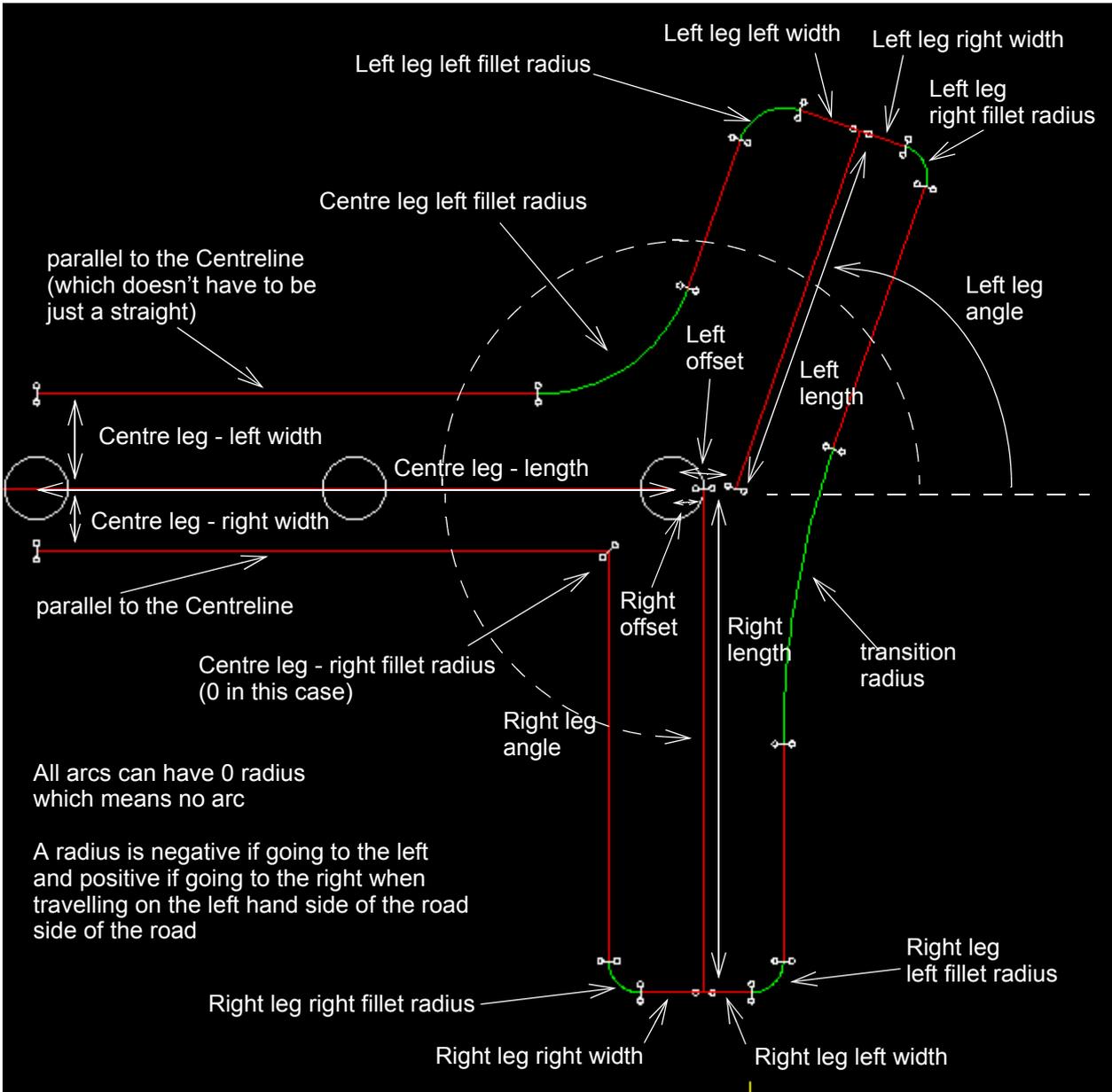
All arc radius values can be zero which means that there is no arc.

A radius is negative if it is going to the left and positive if it is going to the right, when travelling on the left hand side of the road.

Clicking on **Set** stores the current values of the parameters in the function.

Clicking on **Save to library** writes the component to the library with its current values. It is then available in the library to use in the future.

The definitions of all the parameters in the panel are:

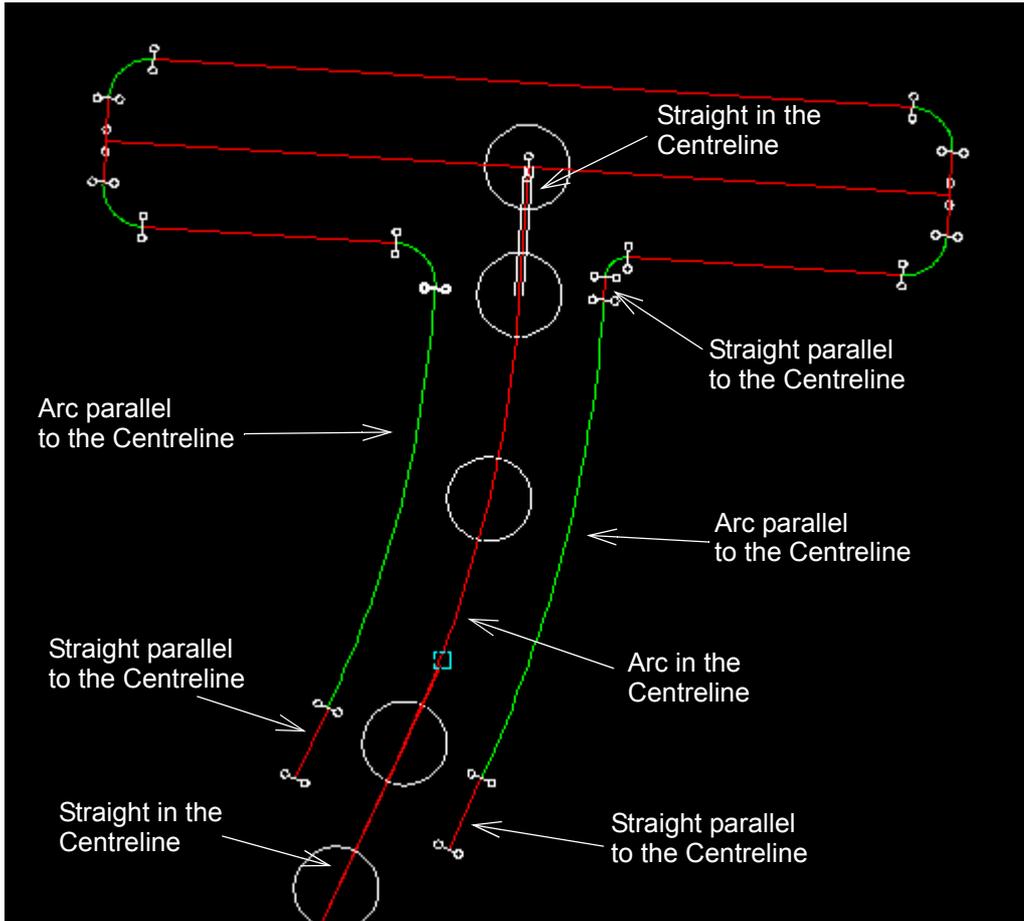


and the parameters that produced this diagram are:

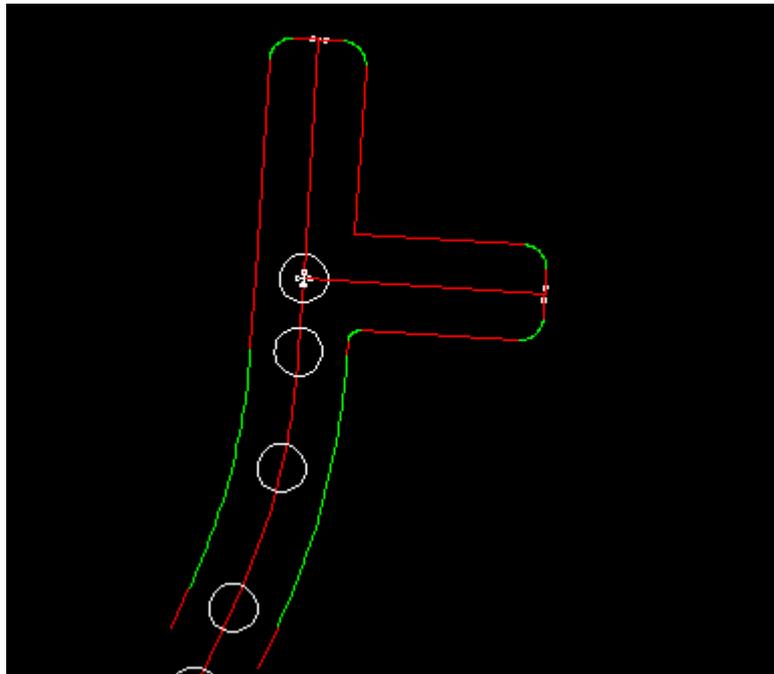
[-] Center leg		
Name	MK\$source_name	abd
Left fillet radius	-5	▲▼
Right fillet radius	0	▲▼
Left width	3	▲▼
Right width	2	▲▼
Length	20	▲▼
[-] Left leg		
Left fillet radius	1.5	▲▼
Right fillet radius	1	▲▼
Left width	3.5	▲▼
Right width	2.5	▲▼
Angle	58	▲▼
Length	12	▲▼
Offset	-1	▲▼
[-] Transition		
Transition radius	-28.5	▲▼
[-] Right leg		
Left fillet radius	1	▲▼
Right fillet radius	1	▲▼
Left width	2.5	▲▼
Right width	3	▲▼
Angle	270	▲▼
Length	17	▲▼
Offset	1	▲▼

**Note** that the Centreline does not have to be a straight. It could involve straights, arcs and transitions.

In the next example, the Hammerhead is coming off a Centreline and the Hammerhead **Centre leg width** value means that the **Centre leg** of the Hammerhead includes a straight, a curve and another straight from the Centreline.



Also note that simply changing the Left leg angle to 0 gives



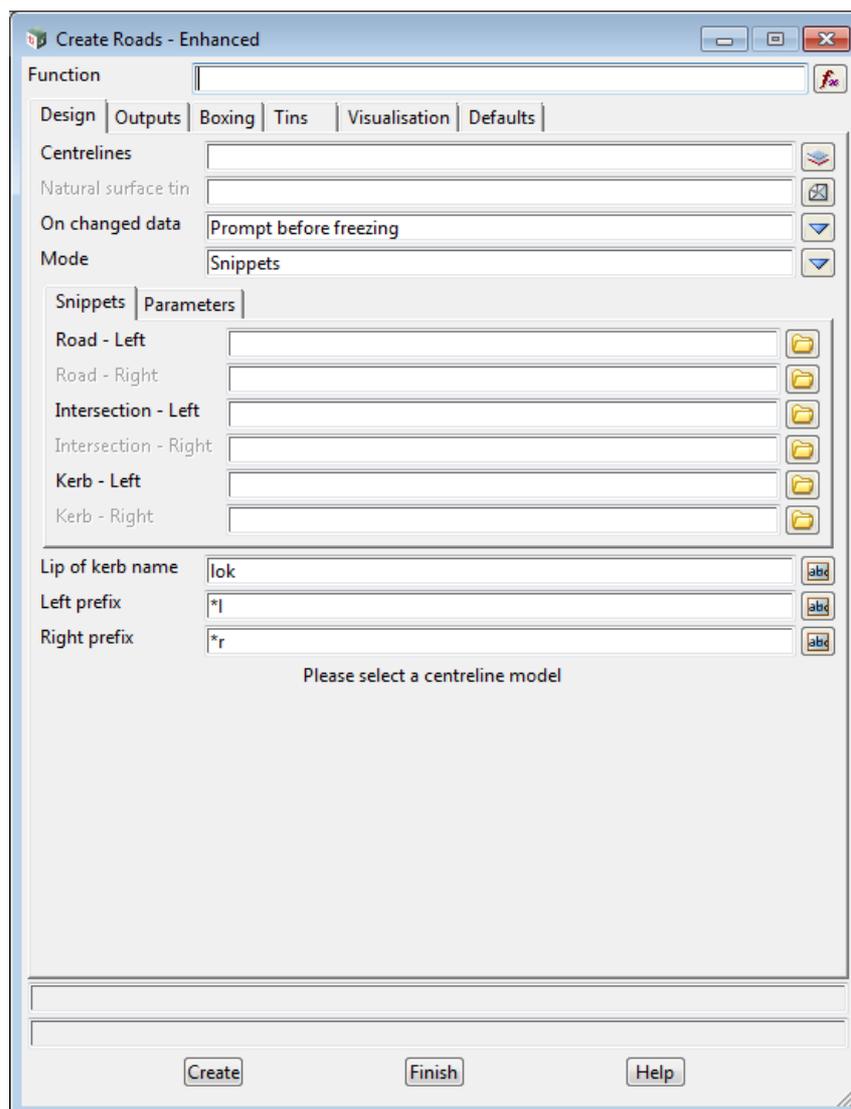
## 22.5.2 Create Roads - Enhanced

**Current position on menu:** Design =>Roads =>Create =>Create Roads - Enhanced

The **Create Roads - Enhanced** panel is a total rewrite of **Create Roads** with many new features including:

- (a) Either *templates* or snippets can be used to define the roads, intersections and kerbs.
- (b) Wherever possible computators have been used to define kerb returns and culdesacs, and Smart Start and End modes in the Apply MTFs rather than chainage values.
- (c) There is now a separate template for the left and right side of the road but unlike **Create Roads**, there is no separate Carriageway template.
- (d) There are default values for culdesacs.
- (e) A kerb can be frozen without freezing the roads.
- (f) Snippets can use the boxing defined in the **Boxing** tab or snippets can do their own boxing. But if you do both then you are on your own.
- (g) Left and Right prefix to use when creating the Apply MTFs.

Selecting **Create Roads - Enhanced** brings up the **Create Roads - Enhanced** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Function</b>	function box		available create roads enhancement functions

*name of the create roads advanced function*

### Available tabs

- Go to [Design tab](#)
- Go to [Outputs tab](#)
- Go to [Boxing tab](#)
- Go to [Tins tab](#)
- Go to [Visualisation tab](#)
- Go to [Defaults tab](#)
- Go to [Buttons at bottom](#)

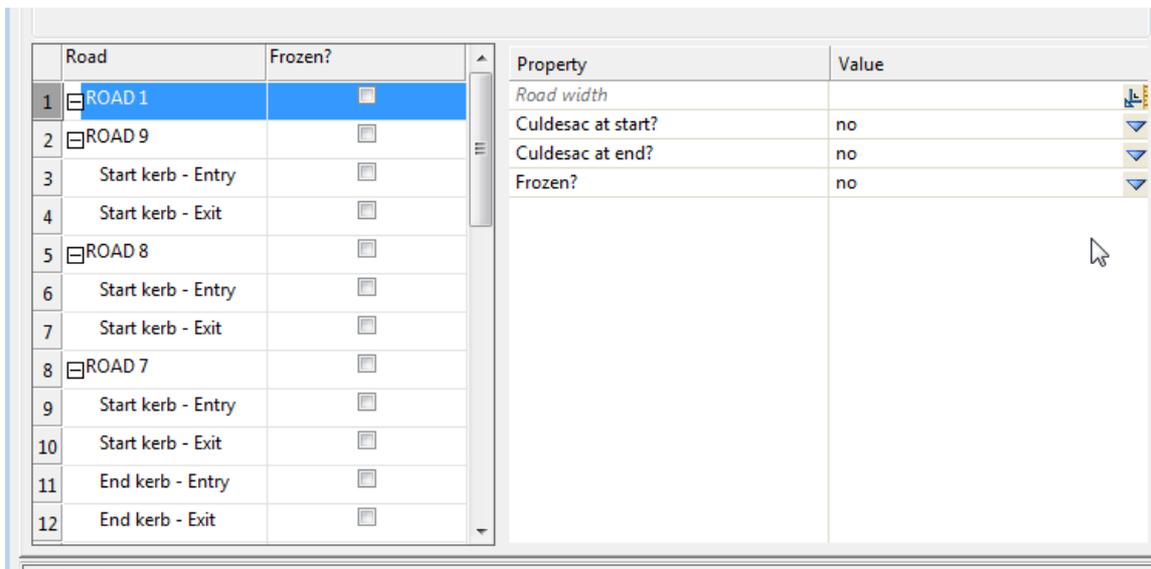
### Design tab

Centrelines	model box	available models
-------------	-----------	------------------

*model of the centreline lines that are to be processed to create roads, intersections and culdesac. Each centreline must have a unique name and vertical geometry.*

*After the Centreline model is selected either from the pop-up or by entering a model name and pushing the <Enter> key, the centrelines are analysed to see if left and right kerbs are possible (centreline ending on another serenely), if a culdesac is possible (end of a centreline) etc and then all the centrelines are listed in a Property Sheet on the bottom of the panel.*

*The entry for each centreline is where you enter values to override the defaults for road width, kerb radius, culdesac parameters etc. This information is then stored **in the function** rather than as attributes on the centreline strings.*



**Road grid** - list of all the centrelines

**Road**

*centreline name*

**Frozen**

tick box

if ticked, the centreline is frozen out of the automatic Create Roads process.

For a frozen centreline, when the **Create Roads** function is run, the Apply MTF for that centreline is deleted and re-created. However any existing Apply MTF for that centreline will be run. Kerb returns are still calculated using the lip line create by running the existing Apply MTF for the frozen centreline. Hence the mtf for a frozen centreline can be modified and the changes will **not** be deleted when the Create Roads function is recalced.

<b>Natural surface tin</b>	<b>tin box</b>	<b>available tins</b>
<i>if <b>not blank</b>, the tin to use as the natural surface for battering in the Applies.</i>		
<i>If <b>blank</b>, no battering is done.</i>		
<b>On changed data</b>	choice box	Prompt before freezing Always freeze Always ignore
<i>if <b>Prompt before freezing</b>,</i>		
<i>If <b>Always freeze</b>,</i>		
<i>If <b>Always ignore</b>,</i>		

<b>Mode</b>	<b>choice box</b>	<b>Templates, Snippets</b>
<i>if <b>Templates</b>, templates are used to define the road.</i>		
<i>If <b>Snippets</b>, snippets are used to define the road.</i>		
<i>For information on the choice <b>Template</b>, goto <a href="#">If Mode is Templates:</a></i>		
<i>For information on the choice <b>Snippets</b>, goto <a href="#">If Mode is Snippets:</a></i>		
<i>For the next tab, goto <a href="#">Outputs tab</a></i>		

**If Mode is Templates:**

**Templates**

<b>Road - left</b>	template box	available templates
<i>the template to use on the left hand side of the centreline. The width to the lip of kerb name is modified to match the given <b>Road width</b>. Through intersection, only the template links out the Lip of kerb name are used so no battering occurs through an intersection.</i>		
<b>Road - right</b>	template box	available templates
<i>the template to use on the right hand side of the centreline. The width to the lip of kerb name is modified to match the given <b>Road width</b>. Through intersection, only the template links out the Lip of kerb name are used so no battering occurs through an intersection.</i>		
<b>Kerb only</b>	template box	available templates
<i>a template with no carriageway part. It is applied to the generated kerb returns</i>		
<b>Lip of kerb name</b>	text box	
<i>name of the lip of kerb link that must exist in the left and the right templates. Modifiers are applied to make the width to the lip of kerb to be the <b>Road width</b>. The kerb returns are created from the lip of kerb on each road, and the start height and grades are taken from the lip of kerb.</i>		
<b>Left/Right prefix</b>	text box	
<i>prefix/postfix (pre*post) to be used as the LHS/RHS prefix in the created Apply MTFs. These are then applied to the left/right template string names. If pretext only, just give the text. If post text is required, precede it by a *</i>		

Return to [Mode choice box Templates, Snippets](#) or go to the next tab [Outputs tab](#)

If Mode is Snippets:

**Snippets tab**

**Road - left** snippet box available snippets

*the snippet to use on the left hand side of the centreline. The width to the lip of kerb name is modified to match the given **Road width**. The snippet is stopped at intersections.*

**Road - right** snippet box available snippet

*the template to use on the right hand side of the centreline. The width to the lip of kerb name is modified to match the given **Road width**. The snippet is stopped at intersections.*

**Intersection - left** snippet box available snippets

*the snippet to use on the left hand side going through intersections. The width to the lip of kerb name is modified to match the given **Road width**.*

**Intersection - right** snippet box available snippets

*the snippet to use on the right hand side going through intersections. The width to the lip of kerb name is modified to match the given **Road width**.*

**Kerb - left** snippet box available snippets

*the snippet that is applied to the generated left hand kerbs.*

**Kerb - right** snippet box available snippets

*the snippet that is applied to the generated right hand kerbs.*

**Lip of kerb name** text box

*name of the lip of kerb link that must exist in the left and the right snippets. Modifiers are applied to make the width to the lip of kerb to be the **Road width**. The kerb returns are created from the lip of kerb on each road, and the start height and grades are taken from the lip of kerb.*

**Natural surface tin** tin box available tins

*if non blank, the tin to use as the natural surface for battering in the Applies. If blank, no battering is done.*

Return to [Mode choice box Templates, Snippets](#) or go to the next tab [Outputs tab](#)

**Outputs tab**

**Apply MTF functions - Roads**

**Apply MTF function stem** text box APPLY \*

*pre\*post text to use around the centreline name to create the Apply MTF function names for the roads along each centreline. The function names will be unique because the centrelines names must be unique within the centrelines model.*

**Road strings stem** text box \* STRS

*pre\*post text to use with the function name in the created Applies for the design string models for each centreline*

**Road sections stem** text box \* XSECS

*pre\*post text to use with the function name and the created Applies for the design cross section models for each centreline*

**Road polygons mode** choice box Model per centreline Model per centreline

One model  
No Polygons

*if **Model for Centreline**, different models are created for the polygons in each Apply MTF.  
If **One model**, all the polygons created by the Applies are placed in one model.  
If **No Polygons**, no polygons are created by the Applies.*

**If Road polygons mode is Model per centreline:**

**Road polygons** text box \* POLYS

*pre\*post text to use with the function name in the created Applies for polygon models for each centreline*

**If Road polygons mode is One model:**

**Road polygon model** model box available models

*when **One model**, the model for all the polygons created by the Applies.*

**Kerb Returns**

**Apply MTF function stem** text box KRET APPLY \*

*pre\*post text to use around the kerb return names to create the Apply MTF function names for the kerb returns. The function names will be unique because the kerb return names are uniquely derived from the centreline names.*

**Naming** choice box Follow Road Names - LHS driving  
Follow Road Names - RHS driving  
Use Numbers  
Use Letters

*method of naming the kerb returns*

**Kerb returns model** model box available models

*model for all the kerb returns*

**Kerb returns Apply MTF strings model** model box available models

*model for all the strings produced by the kerb return Applies*

**Kerb returns Apply MTF sections model** model box available models

*model for all the x-sections produced by the kerb return Applies*

**Apply MTF map file** map file box available map files

*if **not blank**, the name of the map file to use in the created Apply MTFs*

**Output chain** chain box available chain files

*if **not blank**, a chain is created of all the Apply MTFs that are run by the **Create Roads - Advanced** panel **in the order** that they need to be run.*

*Go to the next tab [Boxing tab](#) or return to [Available tabs](#)*

**Boxing tab**

*boxing can be created for all the road and kerb return strings. The boxing definitions must have a common centreline name in them which will be replaced by the name of each of the road/kerb return centrelines.*

**Create boxing**                      tick box

*if ticked, boxing is applied for all the road and kerb returns. The boxing definitions must have a common centreline name in them which will be replaced by the name of each of the road centrelines or kerb returns.*

**Default boxing file**                      file box                                      \*.blf files

*file of boxing file definitions*

**Default boxing CL delimiter**   text box

*common name in each boxing definition which will be replaced by the name of each road/kerb return centreline*

**Model stem for boxing strings**   text box

*pre\*post text to use with the road/kerb return function name for boxing string models*

**Boxing sections**                      text box

*pre\*post text to use with the road/kerb return function name for boxing section models*

**Boxing kerb return sections and strings in one model**   tick box

*if ticked, the strings for all the kerb return Applies are place in one model rather than one model for each kerb return. Similarly, the sections for all the kerb return Applies are placed in the one model. If not ticked, the model names are created in the same way as for the road centrelines. That is, the Model stem for boxing strings and sections are used with the kerb return function name and the layer stems to produce the kerb return boxing strings and sections models.*

**Model name for boxing kerb strings**   model box                      available models

*when only one model (per layer) for the box kerb strings is used - the base name for the model of kerb strings created for each boxing layer. The pre\*post text in the Layer Stem is applied to this name.*

**Model name for boxing kerb sections**   model box                      available models

*when only one model (per layer) for the box kerb sections is used - the base name for the model of kerb sections created for each boxing layer. The pre\*post text in the Layer Stem is applied to this name.*

**Boxing Layer 1-8 in the grid:**

*if non blank, boxing strings and sections will be created for that layer.*

*The last boxing layer is also referred to as the **subgrade** layer.*

**Layer stem**                      text box

*for this boxing layer, the pre\*post text used when naming models for boxing sections and strings.*

**Road - LHS**                      text box

*for this boxing layer, the boxing definition to use on the left hand side of road centrelines*

**Road - RHS**                      text box

*for this boxing layer, the boxing definition to use on the right hand side of road centrelines*

**Kerb return**                      text box

*for this boxing layer, the boxing definition to use on the kerb returns*

**Intersection - LHS** text box

*for this boxing layer, the boxing definition to use on the left hand side of the road though an intersection. The strings and sections are place in the models for the road centreline going through the intersection.*

**Intersection - RHS** text box

*for this boxing layer, the boxing definition to use on the right hand side of the road though an intersection. The strings and sections are place in the models for the road centreline going through the intersection.*

Go to the next tab [Tins tab](#) or return to [Available tabs](#)

## Tins tab

**Create tin** tick box

*if ticked, create a tin for all the roads, kerb returns and culdesacs*

**Road tin** tin box available tins

*name for the tin created from the all the strings and sections*

**Road tin model** model box available models

*model for the Road tin*

**Road tin colour** colour box available colours

*colour of the road tin*

**Subgrade tin** tin box available tins

*if non blank, a subgrade tin of this name is created*

**Subgrade tin model** model box available models

*model for the Subgrade tin*

**Subgrade tin colour** colour box available colours

*colour of the subgrade tin*

**Nulling angle** real box measures pop up

*angle for nulling the triangles*

**Nulling length** real box measures pop up

*length for nulling the triangles*

### Nulling Seed points

*Seed points are used for nulling triangles in the tins*

**Seed X/Y** two column rid

*X/Y coordinate of a nulling seed points.*

**Add** button

*after clicking on **Add**, a seed point is selected from a view and the (X,Y) coordinates are written to the Seed X/Y grid.*

**Remove** button

clicking on **Remove** will delete the highlighted row from the Seed X/Y point grid.

Go to the next tab [Visualisation tab](#) or return to [Available tabs](#)

**Visualisation tab**

**Create visualisation** tick box

if ticked, the road tin is processed for visualisation

**Apply texture map to tin** tick box

if ticked, the texture map is applied to the road tin

**Draw road tin as solid on view** tick box

if ticked, draw the triangles of a tin as solid colour on the view

**Tin colour to use for cut polygons** colour box available colours

colour to use for all the cut polygons

**Tin colour to use for fill polygons** colour box available colours

colour to use for all the fill polygons

**Tin colour to use for transition polygons** colour box available colours

colour to use for all the polygons in neither cut nor fill

Go to the next tab [Defaults tab](#) or return to [Available tabs](#)

**Defaults tab**

**Roads**

**Road width** real box 3.5 measures pop up

width to be used for the left and right side of the road if one hasn't been defined for any centrelines

**Section separation** real box 5 measures pop up

chainage distance to use for the Applies created for the roads along the centrelines

**Chord/Arc tolerance** real box 0.01 measures pop up

chord to arc tolerance to use in the Applies for the centrelines.

**Kerb returns**

**Kerb radius** real box 5 measures pop up

radius to use for filleting between any centrelines that don't have a left or right radius defined

**Section separation** real box 1 measures pop up

chainage distance to use to use for the Applies that are created for the kerb returns

**Chord/Arc tolerance** real box 0.001 measures pop up

chord to arc tolerance to use for the Applies that are created for the kerb returns

**Culdesacs**

<b>Bulb radius</b>	real box	9	measures pop up
<i>radius of the bulb of the culdesac</i>			
<b>Bulb offset</b>	real box		measures pop up
<i>offset of the centre of the culdesac bulb from the centreline.</i>			
<b>Left width</b>	real box	3.5	measures pop up
<i>width of the left hand side of the throat of the culdesac</i>			
<b>Right width</b>	real box	3.5	measures pop up
<i>width of the right hand side of the throat of the culdesac</i>			
<b>Left radius</b>	real box	15	measures pop up
<i>fillet radius of the curve on the left hand side going from the beginning of the culdesac to either the bulb of the culdesac, or onto the tangent between this curve and the bulb of the culdesac.</i>			
<b>Right radius</b>	real box	15	measures pop up
<i>fillet radius of the curve on the right hand side going from the beginning of the culdesac to either the bulb of the culdesac, or onto the tangent between this curve and the bulb of the culdesac.</i>			
<b>Left tangent</b>	real box		measures pop up
<i>if non zero, there is a straight line of this length that is the tangent between the fillet curve on the left hand side at the beginning of the culdesac and the bulb of the culdesac. This can not be a negative value.</i>			
<b>Right tangent</b>	real box		measures pop up
<i>if non zero, there is a straight line of this length that is the tangent between the fillet curve on the right hand side at the beginning of the culdesac and the bulb of the culdesac. This can not be a negative value.</i>			
<b>Projection</b>	real box		measures pop up
<i>if non zero, the end of the centreline with the culdesac on it is extended by this distance. The centre of the culdesac bulb is then placed at the perpendicular distance of <b>Bulb offset</b> from the extended point.</i>			

Go to the next tab [Buttons at bottom](#) or return to [Available tabs](#)

## Buttons at bottom

**Create** button

*run the option and create the road network.*

*To return to the beginning of this option, click on [22.7 Tunnels and Structures](#).*

## 22.6 Sight Lines

Position of menu:   **Design =>Sight lines**  
                          **Utilities =>H-Z =>Sight lines**

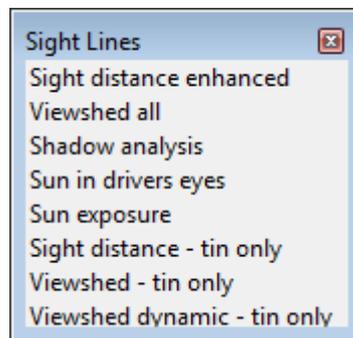
In 12d Model 10, sight lines were only stopped by a tin.

In 12d Model 11, the creation of sight lines has been enhanced in V11 so that most 12d 3d object will stop a sight line. These extra items include

- (i) Super strings that are round pipes or rectangular pipes
- (ii) Billboards
- (iii) Extrusions down a super string
- (iv) trimeshes

The sight line calculations are used in all the options on the Sight Lines menu.

The **Sight Lines** menu is:



See

[22.6.1 Viewshed All](#)

[22.6.2 Shadow Analysis](#)

[22.6.3 Sight Distance Enhanced](#)

## 22.6.1 Viewshed All

Position of option on menu: **Design =>Sight lines =>Viewshed all**  
**Utilities =>H-Z =>Sight lines =>Viewshed all**

This panel is used to calculate the target points, on or above a tin, that are visible and invisible from a user selected eye position. This is equivalent to calculating the target points that can or can not see a selected point.

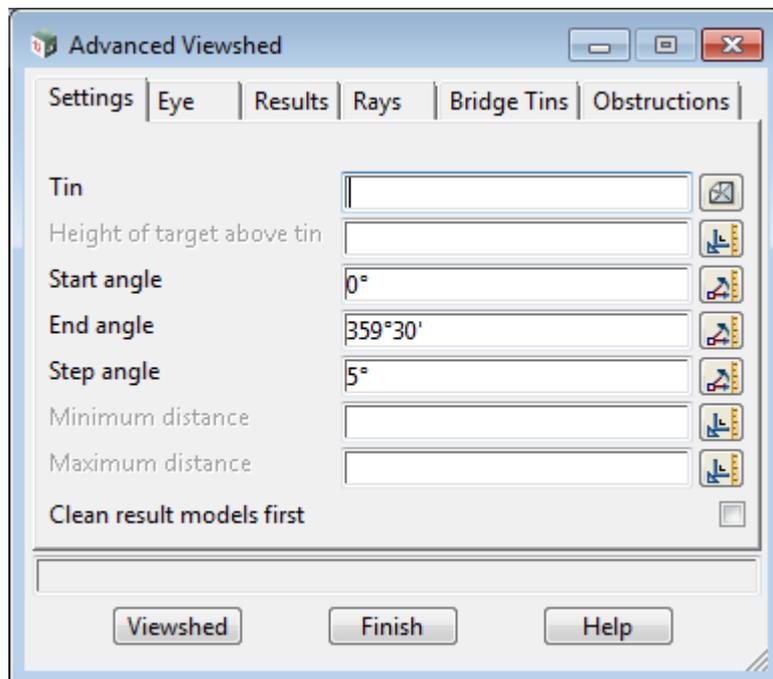
The calculations are made along rays emanating from the eye point from a minimum to a maximum distance from the eye point. The rays are created at regular angular steps from a start angle to an end angle.

Additions for the V11 **Advanced Viewshed** panel are:

- (a) The option allows more items than just one tin to be used in the visibility calculations.
  - (i) Super stings that are round pipes or rectangular pipes are considered in the calculations.
  - (ii) Billboards are considered in the visibility calculations.
  - (iii) Extrusions down a super string are considered in the visibility calculations.

All such super strings in the data source of the **Obstructions** tab are used in the visibility calculations.
- (b) Viewshed will work with null triangles in the tin.
- (c) Viewsheds will work when the eye is off the tin.
- (d) There is a new optional field called **Target height**.
- (e) **Bridge Tins** can also be used as obstructions in the option. A **Bridge Tin** is a tin where the underside of the tin is used in the visibility calculations.

Selecting **Viewshed test** displays the **Advanced Viewshed** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

## Settings tab

- Tin** tin box available tins  
*name of the tin to be used for the viewshed analysis.*
- Height of target** real box  
*the distance that the target is above the tin. This can be positive or negative.*
- Start angle** angle box 0  
*the angle to begin taking sight rays emanating from the eye point. Measured in a counter clockwise direction from the positive x-axis.*
- End angle** angle box 359.30  
*the angle to stop taking sight rays emanating from the eye point. Measured in a counter clockwise direction from the positive x-axis.*
- Step angle** angle box 5  
*the angle between successive sight rays emanating from the eye point.*
- Minimum distance** real box  
*if **not blank**, the minimum distance from the eye point along the sight ray to begin recording visible/invisible points.  
 If **blank**, the minimum distance is zero.*
- Maximum distance** real box  
*the maximum distance from the eye point along the sight ray to record visible/invisible points.  
 If blank, the entire tin is considered.*
- Clean models first** tick box  
*if **ticked**, the models of results are cleaned out before the option runs.*

## Eye tab

- Eye XYZ** input/output xyz ops menu  
**X/Y/Z coordinate**  
*the XYZ co-ordinates of the eye point.  
 Each co-ordinate can be individual entered or the point pick icon used to select an existing point.*
- Note - the z-value must be above the surface of the tin at the (x,y) eye position minus the target height.  
 So the eye can not be below the tin by more than the target height, or even the target height below the tin.*

## Results tab

- More** tick box  
*if **ticked**, ask for name, model, linestyle, weight.  
 If **not ticked**, only ask for model.*
- Name** name box names in names.4d  
*the name for the visible/invisible sight strings.*
- Model** model box available models  
*the model for the visible/invisible sight strings.*

<b>Linestyle</b>	linestyle box	available linestyles
	<i>the linestyle for the visible/invisible sight strings.</i>	
<b>Weight</b>	weight box	
	<i>thickness of the string when plotted</i>	
<b>Colour of visible bits</b>	colour box	default line colour
	<i>if not blank, the colour for the visible parts of the rays. If blank, the visible parts are not created.</i>	
<b>Colour of invisible bits</b>	colour box	default point colour
	<i>if not blank, the colour for the invisible parts of the rays. If blank, the invisible parts are not created.</i>	
<b>Separate strings</b>	tick box	
	<i>if ticked, the visible and invisible sections are separate strings. If not ticked, the visible and invisible sections are combined into one super string.</i>	

## Rays tab

### Visible Rays

*the visible rays are the lines from the eye point to the point on the tin where the target goes from being visible to invisible.*

*If Colour of visible bits is not blank then this is the colour used for these rays.*

*If Colour of visible bits is blank then the colour is the default line colour.*

*The ray will be coloured where it is over invisible targets with the colour for the invisible bits (this can happen in undulating country).*

<b>More</b>	tick box
	<i>if ticked, ask for name, model, linestyle, weight. If not ticked, only ask for model.</i>

<b>Name</b>	name box	names in names.4d
	<i>the name for the visible ray strings.</i>	

<b>Model</b>	model box	available models
	<i>if not blank, the model for the visible ray strings. If blank, don't create the visible ray strings</i>	

<b>Linestyle</b>	linestyle box	available linestyles
	<i>the linestyle for the visible ray strings.</i>	

<b>Weight</b>	weight box
	<i>thickness of the visible ray strings when plotted</i>

### Invisible Rays

*the invisible rays are the lines from the eye point to the point on the tin where the target goes from being invisible to visible.*

*If Colour of invisible bits is not blank then this is the colour used for these rays.*

*If Colour of invisible bits is blank then the colour is the default point colour.*

The ray comes from the eye and is coloured with the **Colour of visible bits** to the change over from visible to invisible and then coloured with the **Colour of invisible bits** to where the ray ends (the change over visible to invisible).

**More** tick box

if **ticked**, ask for name, model, linestyle, weight.  
If **not ticked**, only ask for model.

**Name** name box names in names.4d  
the name for the invisible ray strings.

**Model** model box available models  
if **not blank**, the model for the invisible ray strings.  
If **blank**, don't create the invisible ray strings

**Linestyle** linestyle box available linestyles  
the linestyle for the invisible ray strings.

**Weight** weight box  
thickness of the invisible ray strings when plotted

### Bridge Tins tab

A **Bridge Tin** is a tin where the underside of the tin is used in the visibility calculations.

**Tins** grid of tins available tins

the underside of these tins are used in the visibility calculations. These tins can be used to model some obstructions.

### Obstructions tab

the following elements can also obstruct the view line.

- (i) super strings with round or rectangular sections.
- (ii) bill boards
- (iii) extrusions along a string

**Data source type** Model  
data selection type - XX

**Data source for obstructions**  
source of data to look through for elements to use as obstructions.

**Viewshed** button  
On selecting this button, the lines of sight emanating from the eye XYZ point are calculated.

<Esc> can be used to terminate the option during viewshed calculations.

## 22.6.2 Shadow Analysis

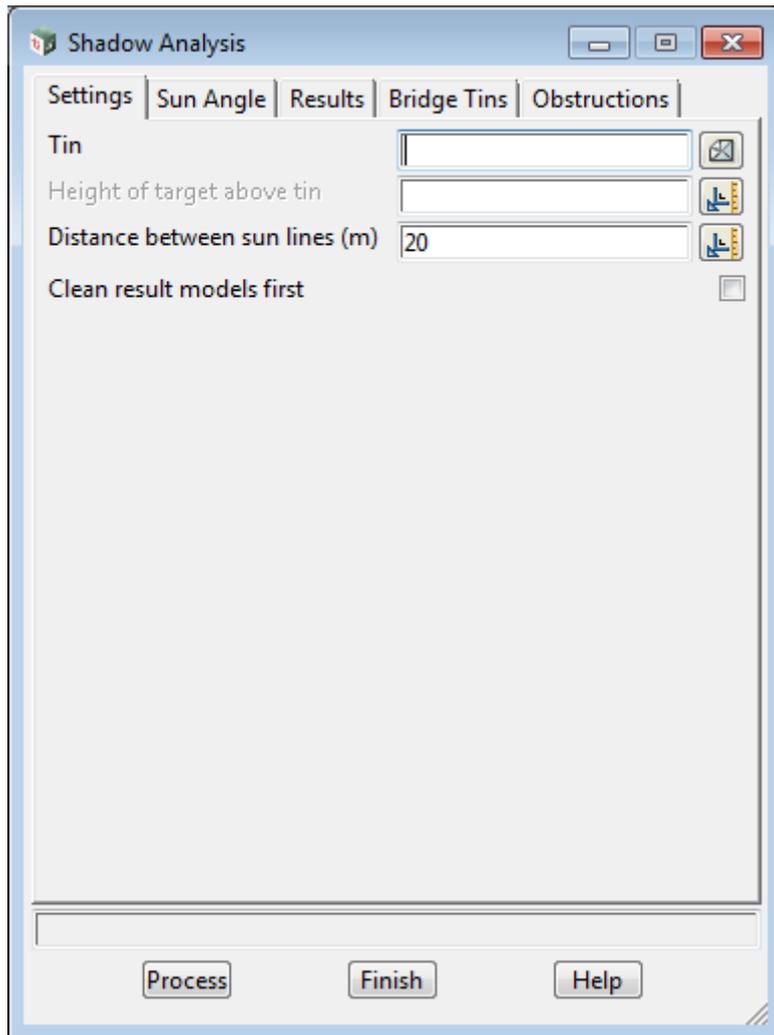
Position of option on menu: **Design =>Sight lines =>Shadow analysis**

**Utilities =>H-Z =>Sight lines =>Shadow analysis**

There is a new **Shadow Analysis** panel.

The **Shadow Analysis** option is similar to the *Viewshed* option except that the **sun** is used as the **eye** position. This means that the **visible** areas are those in **sunlight** and the **invisible** areas are those in the **shade** (in **shadows**).

Selecting **Shadow analysis** displays the **Shadow Analysis** panel.



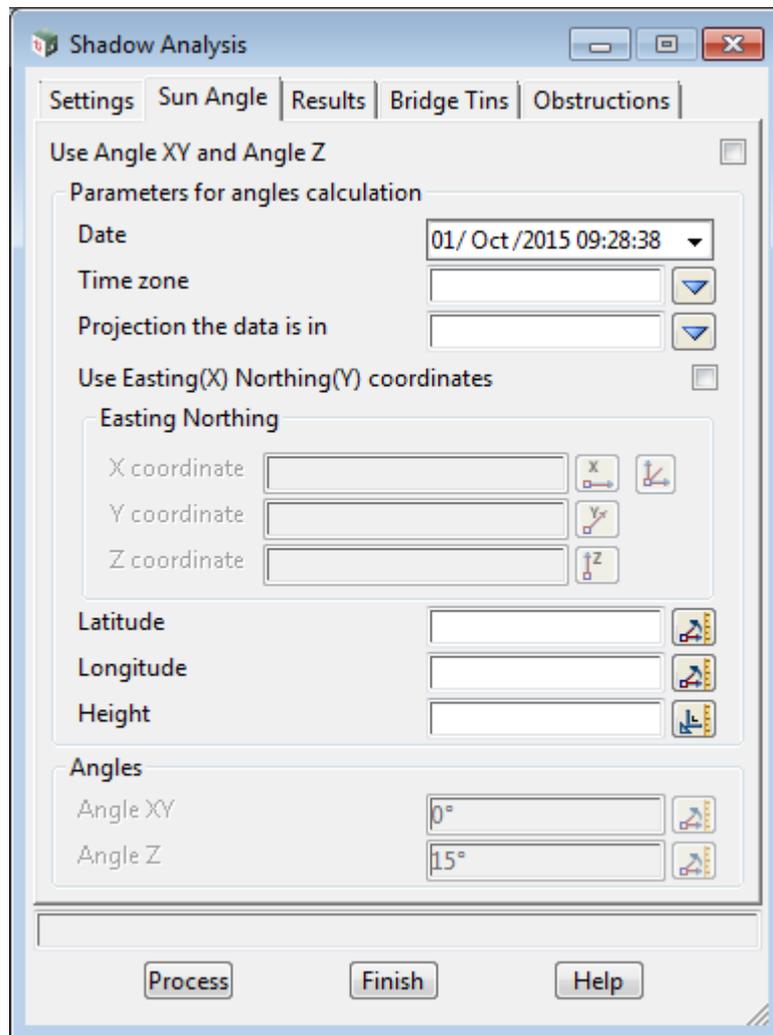
The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tin</b> <i>name of the tin to be used for the shadow analysis.</i>	tin box		available tins
<b>Distance between sun lines</b> <i>the perpendicular distance between the result lines.</i>	real box	20	measures pop up

**Clean models first**      tick box      not ticked

*if ticked, the models of results are cleaned out before the option runs.*

**Sun Angle tab**



**Use Angle XY and Angle Z**      tick box      not ticked

*this controls whether Angle XY and Angle Z are typed in and used to specify the position of the sun, or if the sun position (and hence Angle XY and Angle Z) are calculated by giving the date and time of day, and either the (Easting, Northing) or the longitude and latitude of a point in the middle of the data.*

*If **ticked**, the user specifies the position of the sun with respect to the data by giving **Angle XY** and **Angle Z** (see below for their definitions). This method is used when the data is in an arbitrary coordinate system so there is no way to calculate the position of the sun from the data.*

*If **not ticked**, then the data needs to be in a Projection and the angle to the sun is calculated by giving the date and time, and either the (x,y) coordinate, or the longitude and latitude and of a point in the data.*

So if **Use Angle XY and Angle Z for sun position** is *not* ticked:

**Date**

*the sun position is calculated for this date and time.*

**Time zone** choice box

*used to define the time zone for the time and date.*

**Projection the data is in** projection box

*the data to do Shadow Analysis on must be in this projection.*

**Use Easting (X) Northing (Y) coordinates** tick box not ticked

*if ticked, the (x,y) coordinates of a point in the middle of the data is given. Using the Projection, the longitude and latitude can be calculated for this point. **Note** - because the data is in a projection, then the (x,y) coordinates are often referred to as Eastings and Northings.*

*If not ticked, the latitude and longitude of a point in the middle of the data is given. Using the Projection, the (x,y) position of the point can be calculated.*

**If Use Easting (X) Northing (Y) coordinates is ticked:**

**X/Y/Z coordinate** real box measure box

*the x,y and z coordinates for a typical point in the data can be typed in or a point can be selected using the Point select and the values for x, y and dz will be written to these fields.*

*The z coordinate will also be written to the **Height** field.*

**If Use Easting (X) Northing (Y) coordinates is not ticked:**

**Longitude/Latitude** angle box measure box

*the longitude and latitude for a typical point in the data is typed in.*

**Height** measure box

*height of a typical point on the ground.*

**If Use Easting (X) Northing (Y) coordinates is ticked:**

**Angle XY** angle box 0

*the angle of the sun in the XY plane measured in a counter clockwise direction from the positive x axis. It is dms in HP notation.*

**Angle Z** angle box 15

*the angle of the sun in the vertical plane, measured from the horizontal plane with positive angle going up (Elevation angle?). It is dms in HP notation.*

## Results tab

**More** tick box

*if ticked, ask for name, model, linestyle, weight.  
If not ticked, only ask for model.*

**Name** name box names in names.4d

*the name for the sunshine (visible)/shade (invisible) sight strings.*

**Model** model box available models

*the model for the sunshine (visible)/shade (invisible) sight strings.*

**Linestyle** linestyle box available linestyles

*the linestyle for the sunshine (visible)/shade (invisible) sight strings.*

**Weight** weight box

*thickness of the sunshine (visible)/shade (invisible) sight strings when plotted*



## 22.6.3 Sight Distance Enhanced

Position of option on menu: **Design =>Sight lines =>Sight distance enhanced**  
**Utilities =>H-Z =>Sight lines =>Sight distance enhanced**

The **Advanced Sight Distance** panel allows more items to obstruct than just the surface tin and the bridge tins.

There is now an **Obstructions** tab with a *Data Source* and the following elements in the *Data Source* are now used in the visibility (sight) calculations:

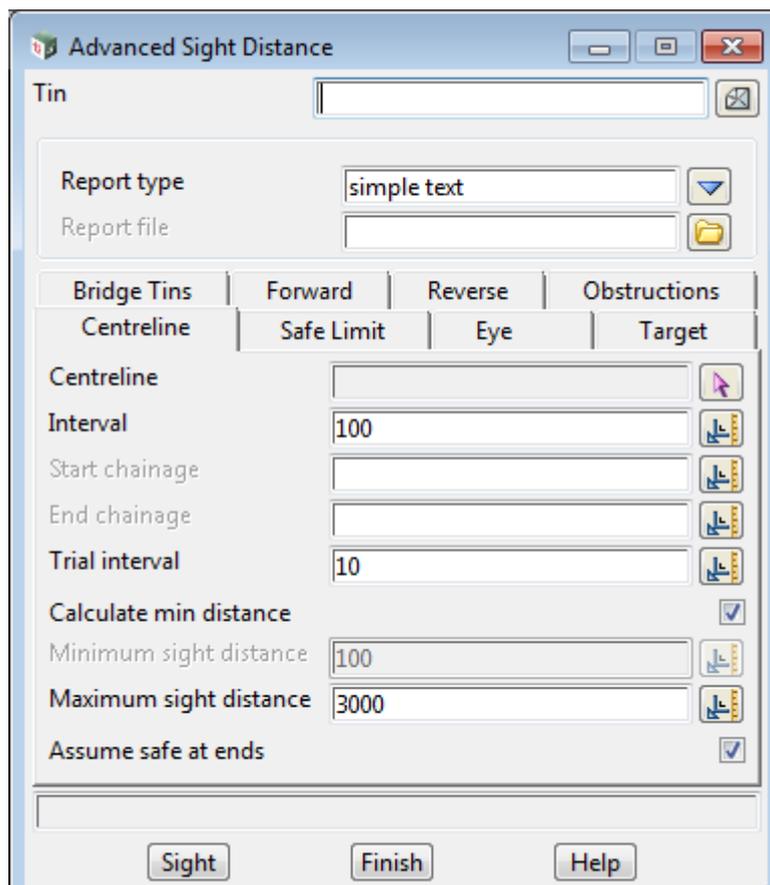
- (i) Super stings that are round pipes or rectangular pipes are considered in the calculations.
- (ii) Billboards are considered in the visibility calculations.
- (iii) Extrusions along a super string are considered in the visibility calculations.

There are also values under the **Safe Limit** tab that can be used in the sight distance calculations instead of the fixed **Minimum sight distance**.

If **Calculate min distance** is ticked, then the values on the **Safe Limit** tab can be entered and are used to calculate a minimum sight distance.

The report is now generated as an XML file number of Report types that can be produced. For example text, pdf, html.

Selecting **Sight Distance Ex** brings up the **Advanced Sight Distance** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tin</b>	tin box		available tins

*name of the tin to use as the surface for testing visibility.*

- Report type** choice box  
*type of report file to be produced (only if **Report file** has a file name in it)*
- Report file** file box depends on Report type  
*if **not blank**, a report of the type selected in **Report type** is produced and written out to this file name.*

**Centreline tab**

- Centreline** string-select  
*the string to be used for calculating the chainage position for the eye and target points, is selected from a view.*
- Interval** real box 100 measures pop-up  
*once the sight distance is calculated for the eye at a chainage, the eye chainage is incremented by this amount the sight distance calculation repeated.*

- Start/end chainage** real box  
*the sight distance is calculated for points on the selected string covering the chainage range given by the start and end chainage fields. If the start/end chainage is blank, the start/end chainage of the selected string is used.*

- Trial interval** real box 10 measures pop-up  
*if the target point is visible, it is then moved along by this chainage increment and the sight test repeated.*

There are also values under the **Safe Limit** tab that can be used in the sight distance calculations instead of the fixed **Minimum sight distance**.

If **Calculate min distance** is ticked, then the values on the **Safe Limit** tab can be entered and are used to calculate a minimum sight distance.

- Calculate min distance** tick box  
*if ticked then the minimum sight distance is calculated from the values in the **Safe Limit** tab which allows for a **Speed, Reaction time** and a **Deceleration coefficient**.  
 If not ticked then the constant **Minimum sight distance** is used.*

- Minimum sight distance** real box 100 measures pop-up  
*If **Calculate min distance** is **not ticked**, then this is the minimum chainage distance from the eye point to place the test target point. The first test of a target is made at this minimum chainage distance from the eye chainage.*

- Maximum sight distance** real box 3000 measures pop-up  
*maximum chainage distance to use for placing the test target point. The testing stops if the test target position goes over this chainage distance. In this case, the sight distance will be the maximum sight distance.*

**Safe Limit tab**

- Speed (km/hr)** real box 60 measures pop-up  
*speed that the vehicle is travelling.*
- Reaction time (s)** real box 2 measures pop-up

*time in seconds for the driver to react.*

**Deceleration coefficient (m/s<sup>2</sup>)**    real box    0.36    measures pop-up  
*the Deceleration coefficient in metres per second per second.*

**Eye tab**

*the eye position is determined by finding the chainage along the selected string, going out perpendicularly for the given eye offset (negative to the left, positive to the right), dropping that position onto the given tin, and then adding the eye height to the height on the tin. Hence the eye point is always the eye height above the tin.*

**Height**    real box    1.3  
*height of the eye point above the given tin.*

**Offset**    real box    -0.5  
*offset of the eye point from the picked string.*

**Target tab**

*the target position is determined by finding the chainage along the selected string, going out perpendicularly for the given target offset (negative to the left, positive to the right), dropping that position onto the given tin, and then adding the target height to the height on the tin. Hence the target point is always the target height above the tin.*

**Height**    real box    0.3  
*height of the target point above the given tin*

**Offset**    real box    -0.5  
*offset of the target point from the picked string.*

**Bridge Tins tab**

*A **Bridge Tin** is a tin where the underside of the tin is used in the visibility calculations.*

**Tins**    grid of tins    available tins

*the underside of these tins are used in the visibility calculations. These tins can be used to model some obstructions.*

**Forward tab**

**Do calcs**    tick box  
*if ticked, do the calculations for the forward direction. That is, from the start of the string and in increasing string chainage.*

**Sight lines**    model box    available models  
*if **not blank**, the name of the model to contain the sight lines.  
 If **blank**, the sight lines are not created.*

**Clean models first**    tick box    not ticked  
*if ticked, the models of results are cleaned out before the option runs.*

**Good colour**    colour box    default colour    available colours  
*when the minimum sight distance is achieved, sight lines are created in the good colour*

**Bad colour**    colour box    default colour    available colours



In the report, the definition for safe distance is

$$\mathit{reaction\_distance} = \mathit{reaction\_time} * \mathit{speed\_value} / 3.6$$

$$\mathit{safe\_distance} = \mathit{reaction\_distance} + \mathit{speed\_value} * \mathit{speed\_value} / (254 * (\mathit{deceleration\_coefficient} + \mathit{grade} * \mathit{change\_deceleration\_per\_grade}))$$

where

**reaction\_time** is in seconds

**speed\_value** is the Operating speed in km/h

**deceleration\_coefficient** is the coefficient of deceleration (longitudinal friction factor)

**grade** is longitudinal grade (a percentage), with sign + for upgrades and - for downgrades.

**change\_deceleration\_per\_grade** is

**Note:** 3.6 is the conversion from *km/h* to *m/s*

## 22.7 Tunnels and Structures

Position of menu: **Design =>Tunnel-Structures**

In *Version 11.0*, the creation of tunnel profiles and the generation of a tunnel or structure has been enhanced from *Version 10.0* to make the process easier.

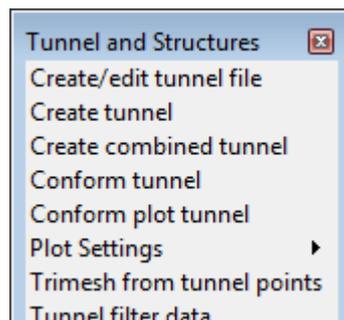
Tunnels and structures are defined by a number of **profiles**, each of which must have the same number of named vertices, and these profiles are applied along a centreline (alignment). New profiles are generated at user given chainages along the tunnel/structure centreline, and any profile generated between two defined profiles is the linear interpolation by chainage of the two defined profiles.

The defined tunnel profiles and how they are applied down a string are specified in the **Create/Edit Tunnel File** panel, and this information is stored in a *Tunnel Definition File* (ending in **.12d\_Tunnel**).

The *Tunnel Definition File* is used in other options such as **Create Tunnel** which generates the profile and longitudinal strings, and trimesh, for the tunnel, **Conform Tunnel** for tunnel conformance, and in **12d Field** for setting out tunnels.

**Note:** a **Trimesh** can be generated for a tunnel by the **Create Tunnel** panel.

The **Tunnel and Structures** menu is:



See

[22.7.1 Creating a Tunnel Definition File](#)

[22.7.2 Create Tunnel](#)

[22.7.3 Conform Tunnel](#)

[22.7.4 Conform Plot Tunnel](#)

[22.7.5 Plot Setting](#)

[22.7.6 Trimesh from Tunnel Points](#)

## 22.7.1 Creating a Tunnel Definition File

Position of option on menu: **Design => =>Tunnel-Structures =>Create/edit tunnel file**

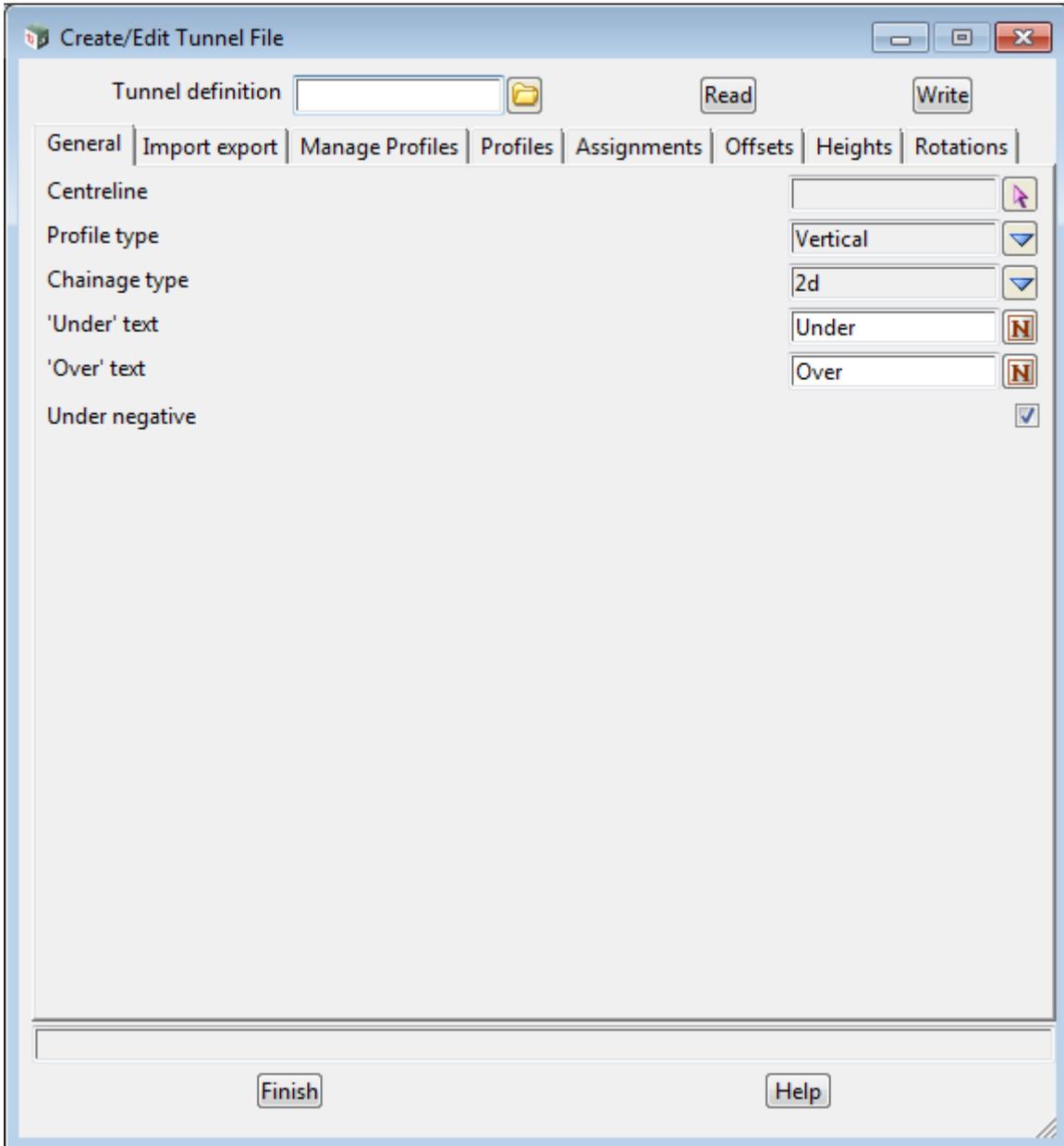
The **Create/Edit Tunnel File** panel is used to define the tunnel definition file which records:

- (a) the profiles used to define the tunnel
- (b) the tunnel centreline (the tunnel reference string) that the tunnel profiles are applied to
- (c) whether the tunnel profiles are applied perpendicular (*i.e.* normal) or vertically to the tunnel centreline
- (d) whether 2d or 3d chainages are used in the tunnel definitions.
- (e) the chainages where the profiles are applied, and optional height, grade and rotation changes when the profiles are applied along the tunnel centreline.

**Note** tunnel profiles and longitudinal strings can be generated perpendicular (*i.e.* normal) to the defined centreline resulting in a true 3d representation of the tunnel.

One restriction is that a tunnel cannot be completely vertical - there must be some chainage difference in the vertical alignment. But given that a chainage difference as small as 0.01mm can be used then this restriction is rarely a problem.

Selecting **Create/edit tunnel file** brings up the **Create/Edit Tunnel File** panel



## Top Section

**Tunnel definition**                      tunnel file box                      available 12d\_tunnel files

*the tunnel file to contain all the definitions for the tunnel. That is, the tunnel centreline, the tunnel profiles and which profiles and how often they are applied along the centreline of the tunnel.*

**Read**    button

*read in the file given in the **Tunnel definition** field.*

**Write**    button

*write out all the values in the panel to the file giving in the **Tunnel definition** field.*

**Export (PRO/PRA)**                      button

*write out the appropriate values in the panel as a PRO and PRA file.*

***Note:** PRO and PRA are files use in TP Setout and TP Stakeout.*

## General tab

### Tunnel centreline

*the centreline that the tunnel definitions are applied to. The centreline must have valid vertical geometry for all the chainage ranges used in the **Assignments**, **Offsets Heights** and **Rotations** tabs.*

**Profile type** choice box perpendicular, vertical

*if **perpendicular**, the tunnel profiles are applied perpendicularly (i.e. normal) to the vertical alignment of the centreline. That is, perpendicular to the tangent vector to the centreline at that chainage. This results in a true 3d model.*

*If **vertical**, the tunnel is calculated vertical to the vertical alignment of the centreline. This may mean an effective loss of clearance on steeper grades.*

**Chainage type** choice box 2d, 3d

*if **2d**, the chainages in the tunnel definitions are taken as plan chainages.*

*if **3d**, are interpreted as 3d. That is, it is the plan/2d chainage of the **1st point** where the horizontal and vertical geometry coincide, plus the 3d length along the centreline from there.*

## Manage Profiles tab

*A tunnel can contain any number of defined profiles that are used to linearly extrapolate between.*

*The **Manage Profile** tab lets you create a new profile:*

*(a) using an existing profile as a seed profile and the new profile is given a copy of the seed profile's geometry. The profile geometry is then edited using the **Profiles** tab.*

*or*

*(b) create a new profile that has no geometry and the geometry is entered using the **Profiles** tab.*

**Seed profile** profiles box available profiles in this tunnel definitions file

*if **not blank**, it must be a profile in this tunnel definitions file and it is used as a Seed profile for the new profile being created.*

**New name** text box

*name for the new profile being created. The new name must be unique amongst the profiles in this tunnel definitions file.*

**Create** button

*create a new profile with the geometry being a copy of the geometry of the Seed profile.*

**Delete profile** profiles box available profiles in this tunnel definitions file

*the profile in this tunnel definitions file to delete.*

**Delete** button

*delete the profile given in the **Delete profile** field.*

**Profile to rename** profiles box available profiles in this tunnel definitions file

*the profile in this tunnel definitions file to be renamed.*

**New name** text box

*new name for the profile. The new name must be unique amongst the profiles in this tunnel definitions file.*

**Rename** button

*rename the profile given in the **Profile to rename** field to the name given in the **New name** field.*

**Import profile** profiles box available profiles in this tunnel definitions file

*the profile in this tunnel definitions file to delete.*

**Delete** button

*delete the profile given in the **Delete profile** field.*

## Profiles tab

*A tunnel can contain any number of defined profiles that are used to linearly extrapolate between.*

*The **Profile** tab lets you create new profiles, pick existing profiles to edit and graphically display the shape of the profile for you.*

**Current Profile** choice box list of profiles in the current tunnel

*Select the profile you wish to create or edit. If the selected profile already has vertices defined, they will be displayed in the grid, and the shape drawn in the area under the grid.*

**Set** button

*Records the contents of the current grid as the profile given in the **Current profile** field. **Set** must be pressed before moving to another profile or the contents of the current grid will be lost.*

## Profile grid

**Name** text box

*the name of the vertex of the profile. It must be unique to this profile.*

**Offset** real box

*the offset of the vertex from the centreline. Positive offset is to the right of the centreline, negative offset is to the left of the centreline.*

**Height** real box

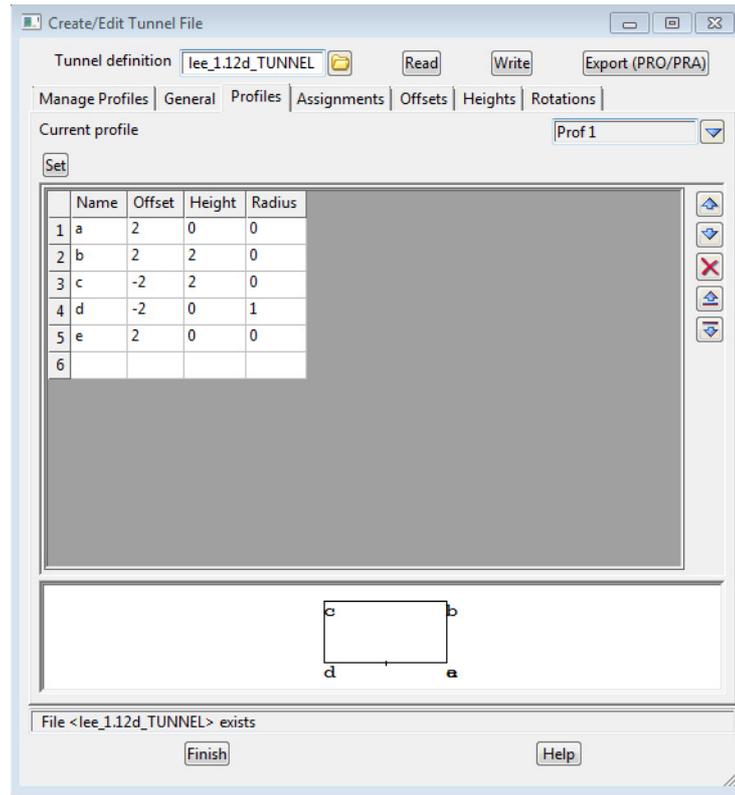
*the height difference of the vertex from the centreline. Positive height is above the centreline, negative height is below the centreline.*

**Radius** real box

*the radius of the segment from this named vertex to the next named vertex. A positive radius is a right handed or clockwise curve with respect to the order of the vertices in the grid.*

*Zero (0) means that there is no radius and there is a straight line between this vertex and the next named vertex.*

*There must always be a value for the radius.*



## Assignments tab

This grid is used to define where the defined profiles are assigned on the centreline. For a row in the grid, the named profile starts at the chainage in that row. For any chainage between this chainage and the chainage on the next row of the grid, a profile is **generated** by interpolating by chainage between the name profile on this row and the named profile on the next row.

The same profile can be used any number of times in the Assignment grid.

The chainages in the Assignments grid must be in ascending order.

**Set** button

Records the contents of the Assignments grid. **Set** must be pressed before moving to another tab or the contents of the current grid will be lost.

## Assignments grid

**Chainage** real cell

The chainage where the named profile begins.

For any chainage between this chainage and the one on the next row, a profile is generated by interpolated by chainage between this profile and the one on the next row of the grid.

**Name** profiles cell available profiles

The name of the profile to start at this chainage.

## Offsets/Heights/Rotations tab

A profile can be adjusted by applying an offset, a (delta) height and a rotation.

The Offsets/Heights/Rotations grid is used to define the offset/height/grade or angle applied to a profile.

The offset/height/grade or angle on a row of the grid starts at the chainage in that row. For any chainage between the chainage of this row and the chainage of the next row, an offset/height/grade or angle for the profile at the chainage is calculated by interpolating between the offset/height/grade or angle of the profile in this row and the offset/height/grade or angle on the following row of the grid. The chainages in the Offsets/Heights/Rotations grid must be in ascending order.

**Notes:**

1. the **height** is a **delta height**. That is, it is added to the height of the profile.
2. the **rotation** in a row can be given as either a **Percent Grade** (positive is up) **or** an **Angle** (dms in hp notation and is measure in counterclockwise from the offset axis). The percent grade or angle will be referred to as the rotation.

**Set** button

Records the contents of the Offsets/Heights/Rotations grid. **Set** must be pressed before moving to another tab or the contents of the current grid will be lost.

**Offsets/Heights/Rotations grid**

**Chainage** real cell

The chainage to start applying the offset/height/rotation.

For any chainage between this chainage and the one on the next row of the grid, an offset/height/rotation is generated by interpolated by chainage between this offset/height/rotation and the one in the next row of the grid.

**Offset/Height/Grade or Angle** real cell

the offset/height/grade or angle to start at this chainage.

For any chainage between this chainage and the one on the next row, an offset/height/rotation is calculated by interpolated by chainage between this offset/height/rotation and the one on the next row of the grid.

**Type** choice cell None, Linear, Cubic  
Rev C, Biquadratic

the type of interpolation used for the offset/height/rotation at chainage between the chainage of this row and the chainage on the next row.

**None:** no interpolation is done and the offset/height/ at this chainage is used. If there is a different offset/height/rotation on the next row then there will be a problem and you need to insert another row after this row with a chainage slightly less than the chainage on the next row. The offset/height/rotation of the inserted row needs to be the same as this row and the Type should be **Linear**.

**Linear:** offset/height/rotation is linearly interpolated by chainage between the offset/height/rotation on this row and the offset/height/rotation on the next row.

**Cubic:** offset/height/rotation is interpolated as a cubic by chainage between the offset/height/rotation on this row and the offset/height/rotation on the next row.

**Rev C:** offset/height/rotation is interpolated as a reverse cubic by chainage between the offset/height/rotation on this row and the offset/height/rotation on the next row.

**Biquadratic:** offset/height/rotation is interpolated as a back to back quadratics by chainage between the offset/height/rotation on this row and the offset/height/rotation on the next row.

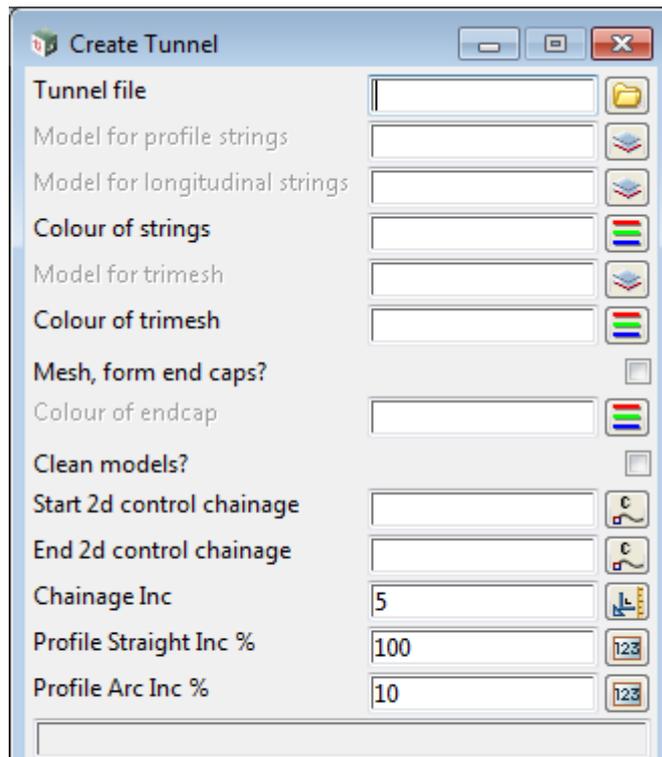
## 22.7.2 Create Tunnel

Position of option on menu: **Design =>Tunnel-Structures =>Create Tunnel**

The **Create Tunnel** panel is used to generate the strings defining the shape of the tunnel - the profile and longitudinal strings, and the trimesh, for the tunnel.

The tunnel is defined in a **Tunnel File** that has been created in the **Create/Edit Tunnel File** panel.

Selecting **Create Tunnel** brings up the **Create Tunnel** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tunnel file</b>	tunnel file box		available 12d_tunnel files
<i>the tunnel file containing all the definitions for the tunnel. That is, the tunnel centreline, the tunnel profiles and which profiles and how often they are applied along the centreline of the tunnel.</i>			

<b>Model for profile strings</b>	model box		
<i>if <b>not blank</b>, a string of the tunnel profile is created at each chainage point along the tunnel centreline and stored in this model.</i>			
<i>If <b>blank</b>, no strings of the tunnel profiles are created.</i>			

<b>Model for longitudinal strings</b>	model box		
<i>if <b>not blank</b>, strings are created by joining common vertices of the profiles along the tunnel centreline and stored in this model.</i>			
<i>If <b>blank</b>, no longitudinal strings are created.</i>			

<b>Colour of strings</b>	colour box		available colours
--------------------------	------------	--	-------------------

*colour of the profiles and longitudinal strings.*

**Model for trimesh**                      model box

*if **not blank**, a trimesh is created for the tunnel and placed in this model.  
If **blank**, no trimesh is created.*

**Colour of trimesh**                      colour box                                      available colours

*colour of the trimesh.*

**Mesh, form end caps?**              tick box

*if **ticked**, the trimesh includes a mesh over each end of the tunnel. This creates a closed trimesh. A volumes is calculated and reported in a String inquire on a closed trimesh.*

*If not **ticked**, the trimesh ends are left open. A volumes is NOT reported in a String inquire on an open trimesh.*

**Clean models?**                              tick box

*if **ticked** the profile and longitudinal strings, and trimesh models are cleaned before the new strings and trimesh are created.*

**Start/End 2d control chainage**              real box

*the Start/End 2d chainage on the centreline to create the trimesh, longitudinal strings and profile strings for.*

**Chainage Inc**                              real box

*the chainage increment to create the sections and vertices of the strings at. Whether the chainage is 2d or 3d is defined in the Tunnel Definition File.*

**Profile Inc%**                              real box

*this defines how often vertices are generated around the tunnel profiles between the vertices of the tunnel profile, and hence how many vertices are created in the sections, how many longitudinal strings and the density of triangles in the trimesh.*

*The value is a percentage to the length of the tunnel profile and for each tunnel section, vertices are generated at each of the defining vertices of the tunnel profile, and then at the **Profile Inc%** distance between adjacent defining vertices of the tunnel profile.*

*Hence **Profile Inc%** controls the accuracy of the shapes generated for the tunnel.*

***Note** - the value **Profile Inc%** give a good approximation to the number of vertices are created on each section but there may be a few more because all the defining vertices of the profile as in the sections. So if **Profile Inc%** is 5 then there will be approximately 20 vertices on each section.*

**Create**                                      button

*create the tunnel profile strings, longitudinal strings, and trimesh.*

## 22.7.3 Conform Tunnel

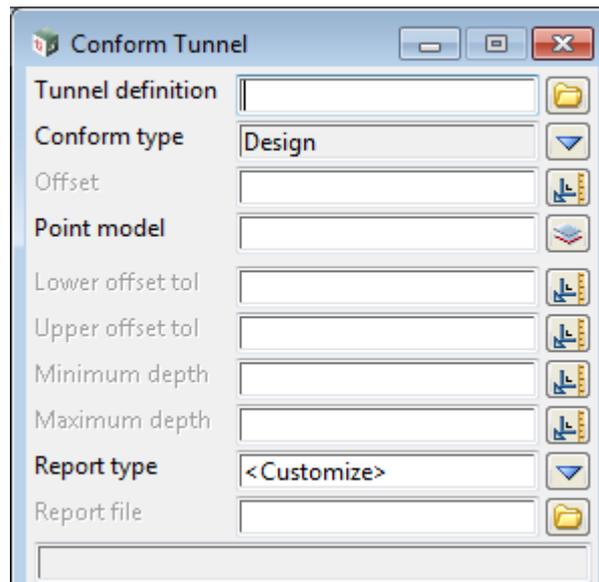
Position of option on menu: **Design =>Tunnel-Structures =>Conform tunnel**

CURRENTLY A WORK IN PROGRESS

The **Conform Tunnel** panel creates a conformance report between points and a tunnel.

The tunnel is defined in a **Tunnel definition** file that has been created in the **Create/Edit Tunnel File** panel.

Selecting **Conform tunnel** brings up the **Conform Tunnel** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tunnel definition</b>	tunnel file box		available 12d_tunnel files <i>the tunnel definition file containing all the definitions for the tunnel. That is, the tunnel centreline, the tunnel profiles and which profiles and how often they are applied along the centreline of the tunnel.</i>
<b>Point model</b>	model box		
<b>Write</b>	button		<i>create the tunnel conformance report.</i>

## 22.7.4 Conform Plot Tunnel

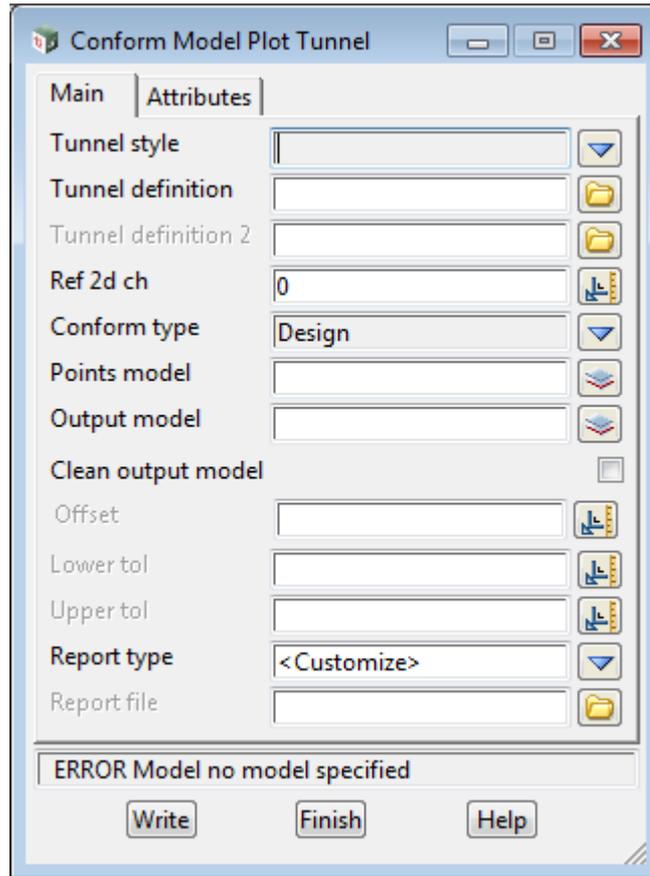
Position of option on menu: **Design =>Tunnel-Structures =>Conform plot tunnel**

CURRENTLY A WORK IN PROGRESS

The **Conform Model Plot Tunnel** panel creates a conformance plot for a tunnel.

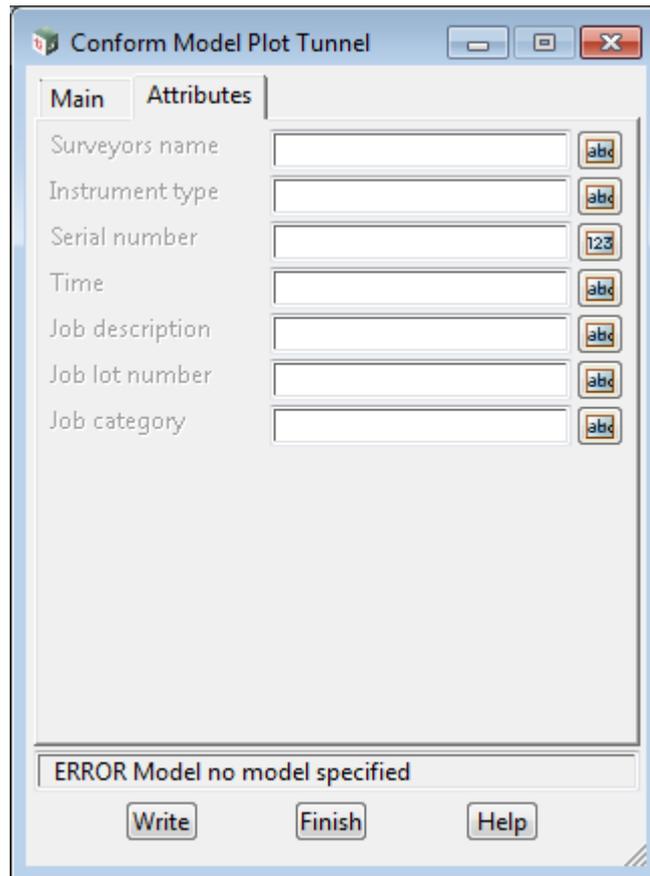
The tunnel is defined in a **Tunnel definition** file that has been created in the **Create/Edit Tunnel File** panel.

Selecting **Conform plot tunnel** brings up the **Conform Model Plot Tunnel** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tunnel definition</b>	tunnel file box		available 12d_tunnel files <i>the tunnel definition file containing all the definitions for the tunnel. That is, the tunnel centreline, the tunnel profiles and which profiles and how often they are applied along the centreline of the tunnel.</i>
<b>Point model</b>	model box		



**Write** button  
*create the tunnel conformance plot.*

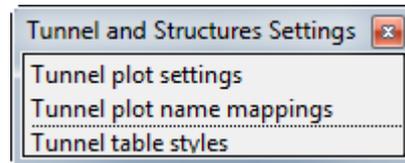
## 22.7.5 Plot Setting

Position of menu: **Design =>Tunnel-Structures =>Plot settings**

CURRENTLY A WORK IN PROGRESS

The **Plot Settings** options define the look of the tunnel conformance plot.

The **Plot Settings** menu is:



See

[22.7.5.1 Tunnel Plot Settings](#)

[22.7.5.2 Tunnel Plot Name Mappings](#)

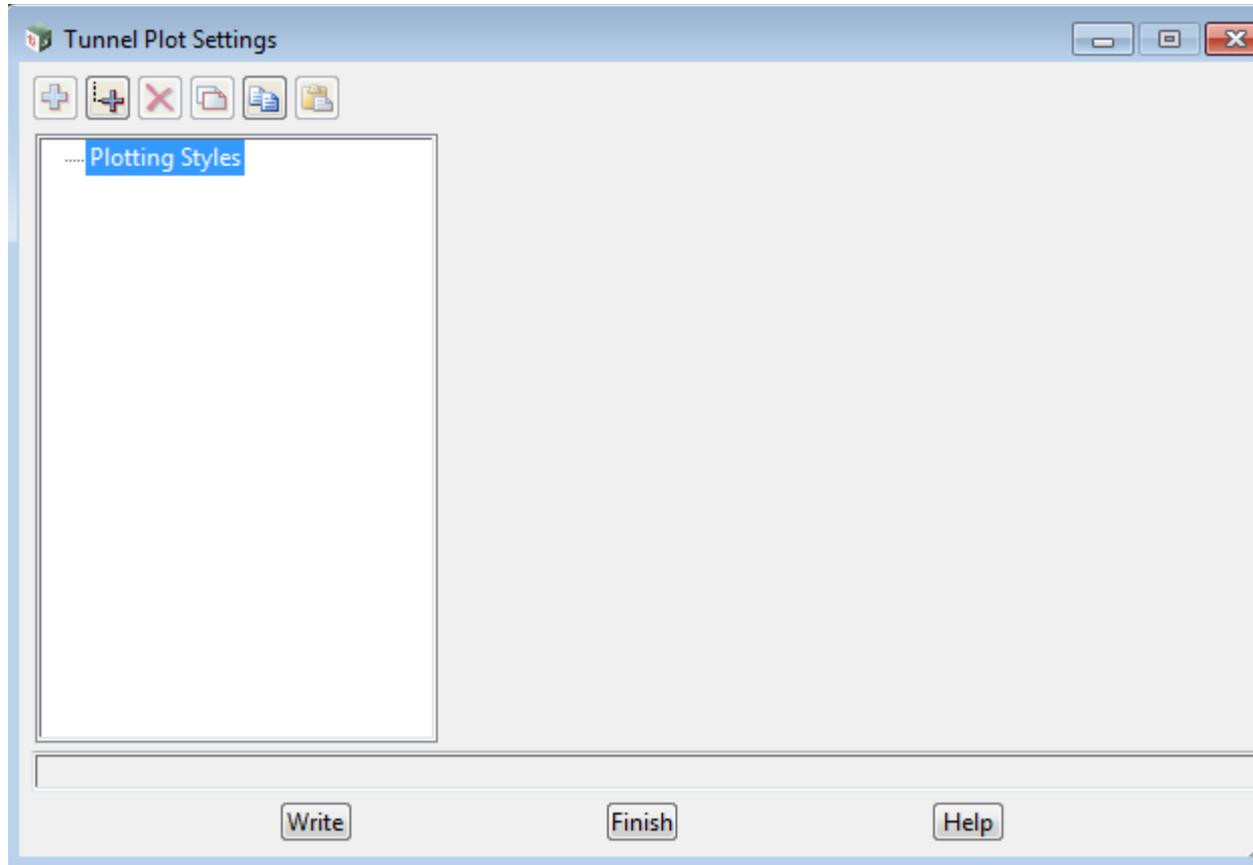
[22.7.5.3 Tunnel Table Styles](#)

## 22.7.5.1 Tunnel Plot Settings

Position of option on menu: **Design =>Tunnel-Structures =>Plot settings =>Tunnel plot settings**

CURRENTLY A WORK IN PROGRESS

Selecting **Tunnel plot settings** brings up the **Tunnel Plot Settings** panel



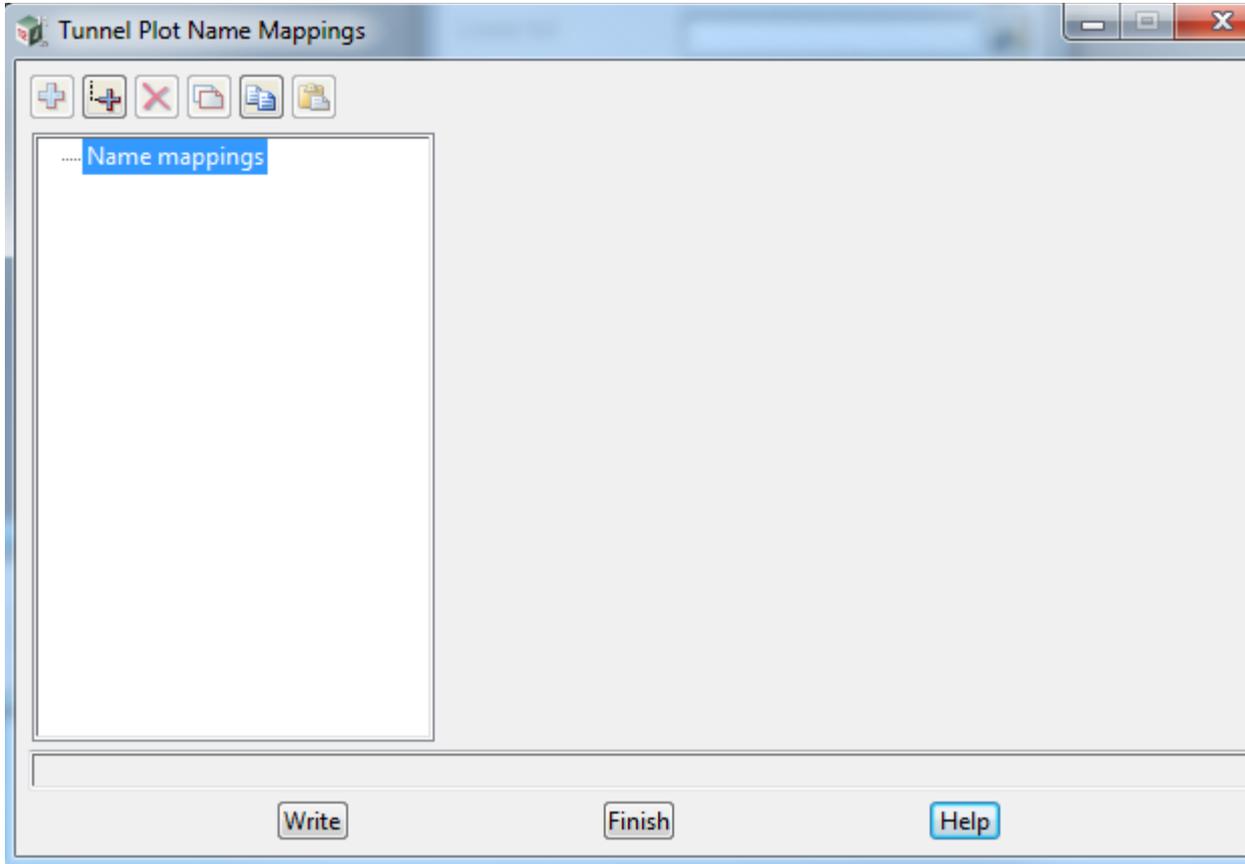
## 22.7.5.2 Tunnel Plot Name Mappings

Position of option on menu:

**Design =>Tunnel-Structures =>Plot settings =>Tunnel plot name mappings**

CURRENTLY A WORK IN PROGRESS

Selecting **Tunnel plot name mappings** brings up the **Tunnel Plot Name Mappings** panel



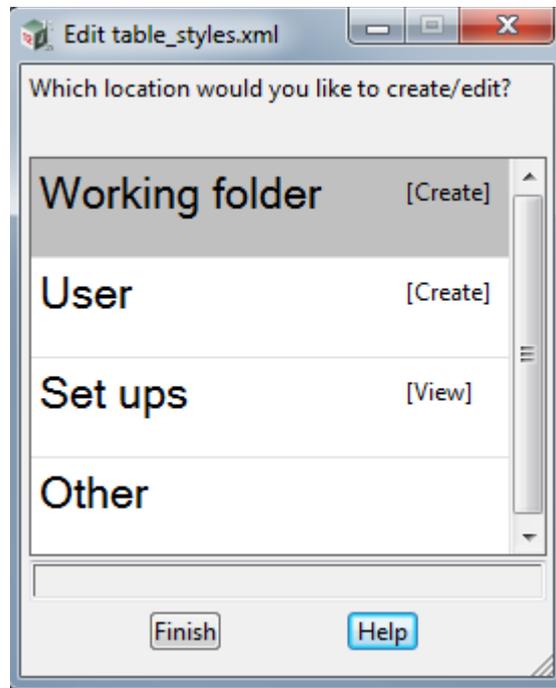
### 22.7.5.3 Tunnel Table Styles

Position of option on menu:

**Design =>Tunnel-Structures =>Plot settings =>Tunnel table styles**

CURRENTLY A WORK IN PROGRESS

Selecting **Tunnel table styles** brings up the **Edit tables\_styles.xml** panel



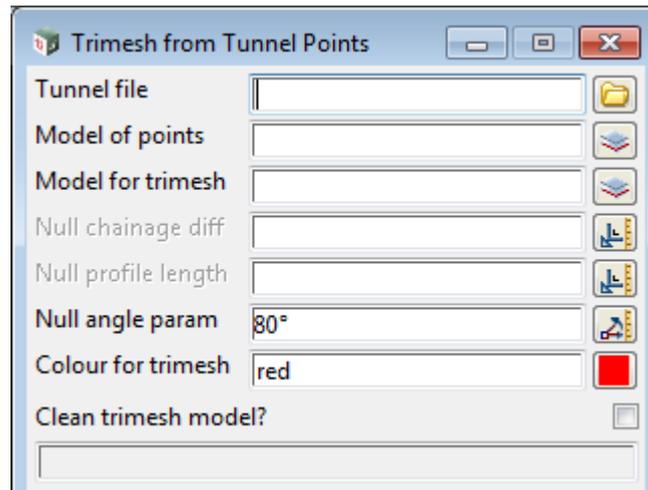
## 22.7.6 Trimesh from Tunnel Points

Position of option on menu: **Design =>Tunnel-Structures =>Trimesh from tunnel points**

**Trimesh from tunnel points** takes points that are usually survey shots of the surface of the tunnel, and creates a trimesh from them. So the points are approximately on the tunnel surface.

The tunnel is defined in a **Tunnel File** that has been created in the **Create/Edit Tunnel File** panel.

Selecting **Trimesh from tunnel points** brings up the **Trimesh from Tunnel Points** panel



The fields and buttons used in this panel have the following functions.

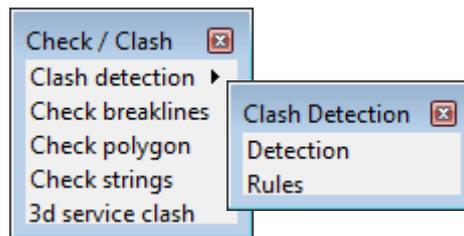
Field Description	Type	Defaults	Pop-Up
<b>Tunnel file</b> <i>the tunnel file containing all the definitions for the tunnel. That is, the tunnel centreline, the tunnel profiles and which profiles and how often they are applied along the centreline of the tunnel.</i>	tunnel file box		available 12d_tunnel files
<b>Model of points</b> <i>the model of points that are approximately on the tunnel surface.</i>	model box		available models
<b>Model for trimesh</b> <i>model for the created trimesh.</i>	model box		available models
<b>Null element param</b>	real box	0.7	
<b>Colour for trimesh</b> <i>colour of the trimesh.</i>	colour box		available colours
<b>Clean trimesh model?</b> <i>if ticked the model for the trimesh is cleaned before the new trimesh is created.</i>	tick box		
<b>Create</b> <i>create the trimesh of the points</i>	button		

## 22.8 Check/Clash

Position of menu: **Design =>Check/clash**  
**Utilities =>A-G =>Check/clash**

A new clash detection walk-right menu, **Clash detection**, has been added to the **Check/clash** menu.

The **Check/Clash** menu is:



For the new options, see

*Detection* [22.8.1 Clash Detection](#)

*Rules* [22.8.2 Clash Detection Rules](#)

## 22.8.1 Clash Detection

**Position of option on menu:** Design =>Check/clash => Clash detection =>Detection  
 Utilities =>A-G =>Check/clash => Clash detection =>Detection

Service Clash detection checks whether strings clash with other strings.

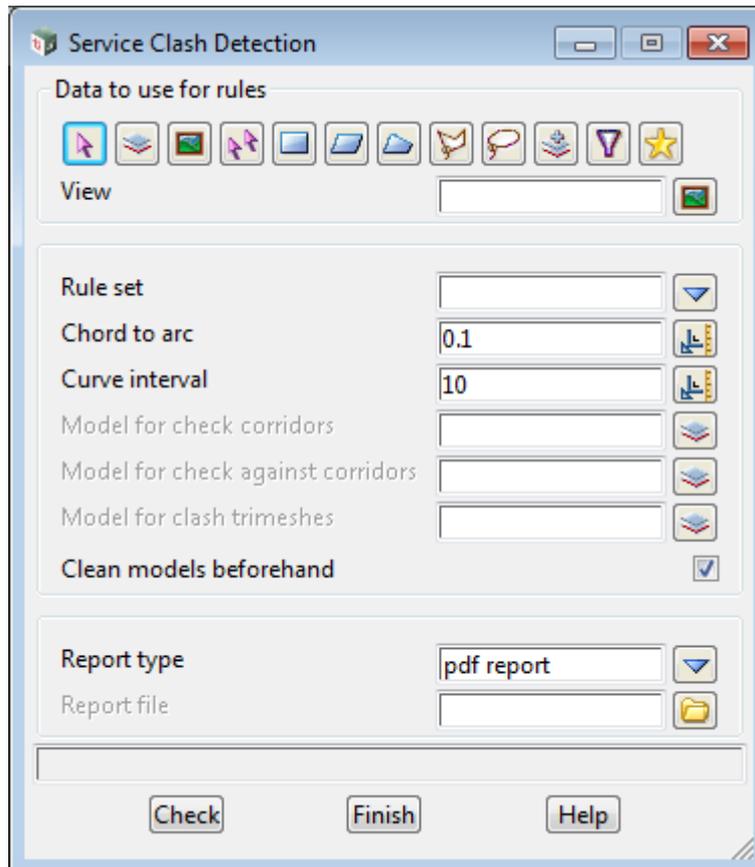
**Rules** can be defined specifying what is a clash. For example, you may only want to know about a horizontal clash, but not a vertical clash, or you may want to know if anything clashes not only with a string but within a user given corridor around the string.

Clashes can be detected between strings of the types Super Pipe with straight line segments and Drainage and Sewer stings with straight line segments.

Trimeshes of the clash corridors around the strings can be created as well as trimeshes of the zones where clashes occur.

The data to feed though to the Rules is selected by a Data Source **Data to user for rules** so that the one Rule set can be used for many different selection sets.

Selecting **Detection** brings up the **Service Clash** panel



The fields and buttons used in the panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Data to Use for rules:**

*the Data Source is used to restrict the data that is to be processed by the Rule set. This means that one set of Rules can be used by many different sets of data by just changing the Data Source and not having to change to Rule set.*

*This panel can be recorded in a Chain.*

**Data source type** Model  
*data selection type - for a full description go to [4.19.3 Data Source](#) in the chapter [4 Tools and Concepts](#)*

**Data source** input  
*source of data to be processed by the **Rule set**.*

**Rule set** choice box available clash rules  
*the Rule Set to use in the clash detection for the data in the Data Source **Data to user for rules**.*

*The Rules in a Rules Set define which models and strings are checked against which models and strings.*

*The pop up displays a list of each of the Rule Sets that are available and one Rule Set is selected to use for this run. For information on defining Rules and Rule Sets. See [22.8.2 Clash Detection Rules](#).*

**Curve interval** real box 10  
*the interval to break curves up when calculating clashes.*

**Chord to arc** real box 0.1  
*the chord to arc tolerance to use when breaking curves up when calculating clashes.*

**Model for check corridors** model box  
*if **not blank**, for all strings in the **Models to check** given in the Rule Set, strings will be created representing the defined corridor.*

**Model for check against corridors** model box  
*if **not blank**, for all strings in the **Models to check against** given in the Rule Set, strings will be created representing the defined corridor.*

**Model for clash trimeshes** model box available models  
*if **not blank**, trimeshes are created to indicate where the clashes occur, and are placed in this model.*

**Clean models beforehand** tick box  
*if **ticked**, the models for check corridors, check against corridors and clash trimesh models are cleaned before processing.*

**Report type** choice box available xml formats for this panel  
*the type of report to be generated from the xml file.  
 New customised report formats can be added by the user. See [7. Setting Up XML Reports](#)*

**Report file** file box  
*if **not blank**, a report file of this name is created.  
 If **blank**, no report is created.*

**Check** button  
*perform the clash checks. Error messages are written to the Output Window as Intelligent Log Lines to help find the problems.*

## 22.8.2 Clash Detection Rules

**Position of option on menu:** Design =>Check/clash => Clash detection =>Rules  
 Utilities =>A-G =>Check/clash => Clash detection =>Rules

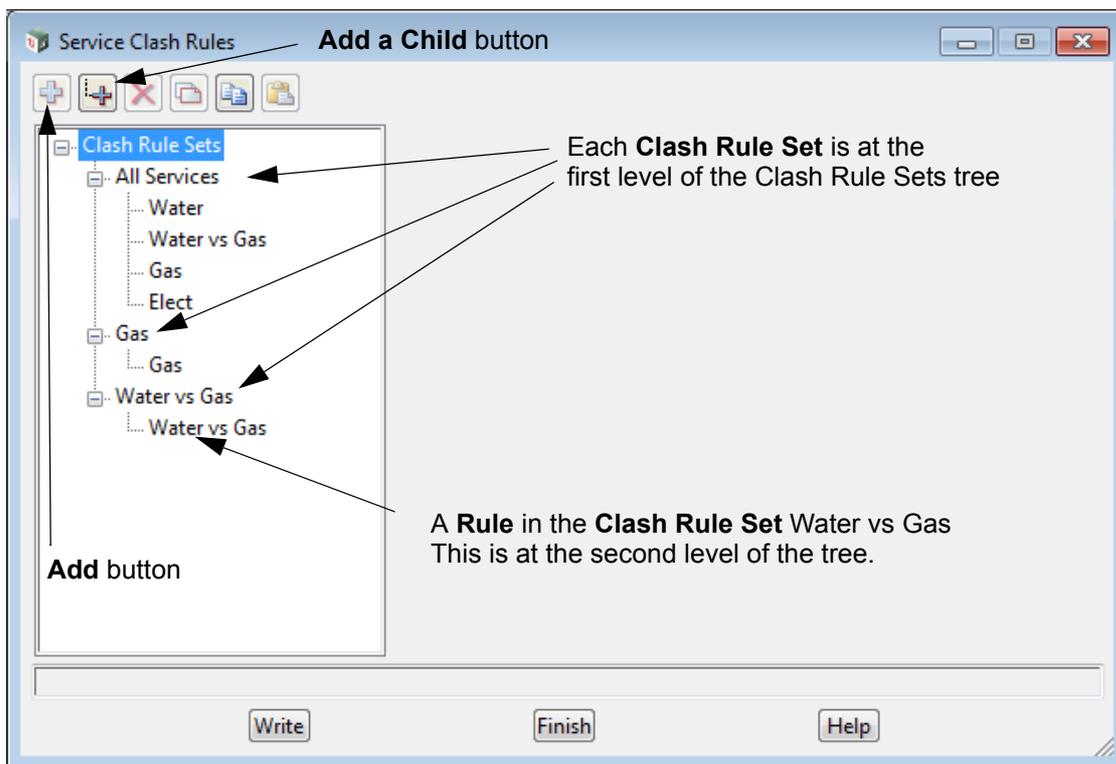
The Service Clash Rules define what is meant by a "clash". They are used in the panel **Service Clash Detection** (see [22.8.1 Clash Detection](#)).

The Service Clash Rules is a tree of *Clash Rule Sets* and each set has a unique name.

The **Service Clash Detection** panel specifies which of the **Clash Rule Sets** is to be used for a particular clash detection run.

The **Service Clash Rules** are loaded from a file called **service\_clash\_rules.xml** that is searched for as a standard **Set\_Ups** file. See [41.2 Files for Setting Up 12d](#).

Selecting **Rules** brings up the **Service Clash Rules** panel



### Write Button at Bottom

When the **Write** button is selected, a **Write Setup File** panel comes up to specify where the **services\_clash\_rules.xml** file is to be written out to.

A project restart is required for the new file to take effect

### Clash Rule Sets Tree

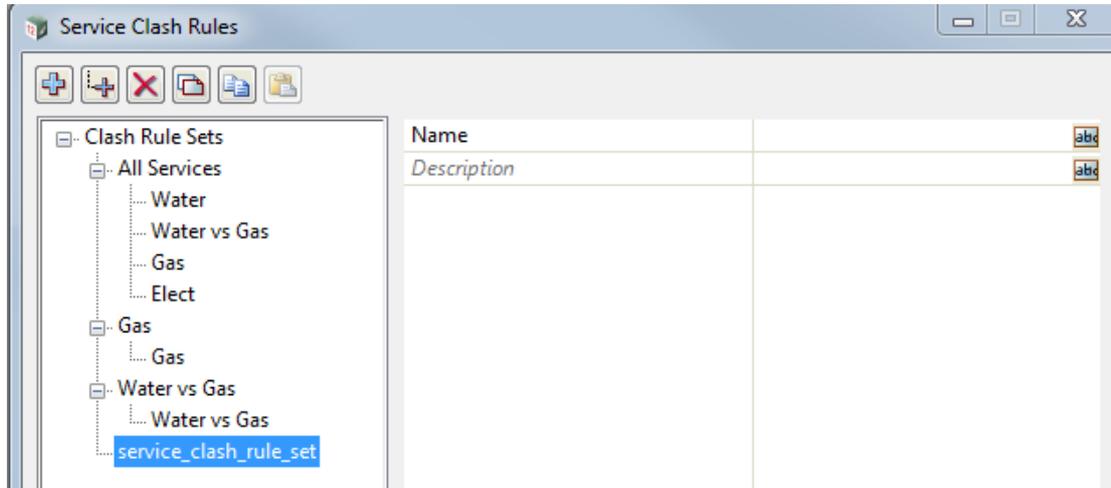
(a) Clash Rule Set

A **Clash Rule Set** is defined at the first level of the **Clash Rule Sets** tree and each **Clash Rule Set** must have a unique name. One **Clash Rule Set** is selected in the **Service Clash Detection** panel to provide the rules to be used for a particular clash detection run

A new *Clash Rule Set* is created by clicking and highlighting the top level *Clash Rules Sets* and then pressing the **Add Child** button, or by clicking on and highlighting a first level *Clash Rule Set* and then pressing the **Add** button.

A new *Clash Rules Set* with the dummy name **service\_clash\_rule\_set** is then created and

a new name must be entered in the **Name** field on the right hand side of the tree. The **Description** is optional.

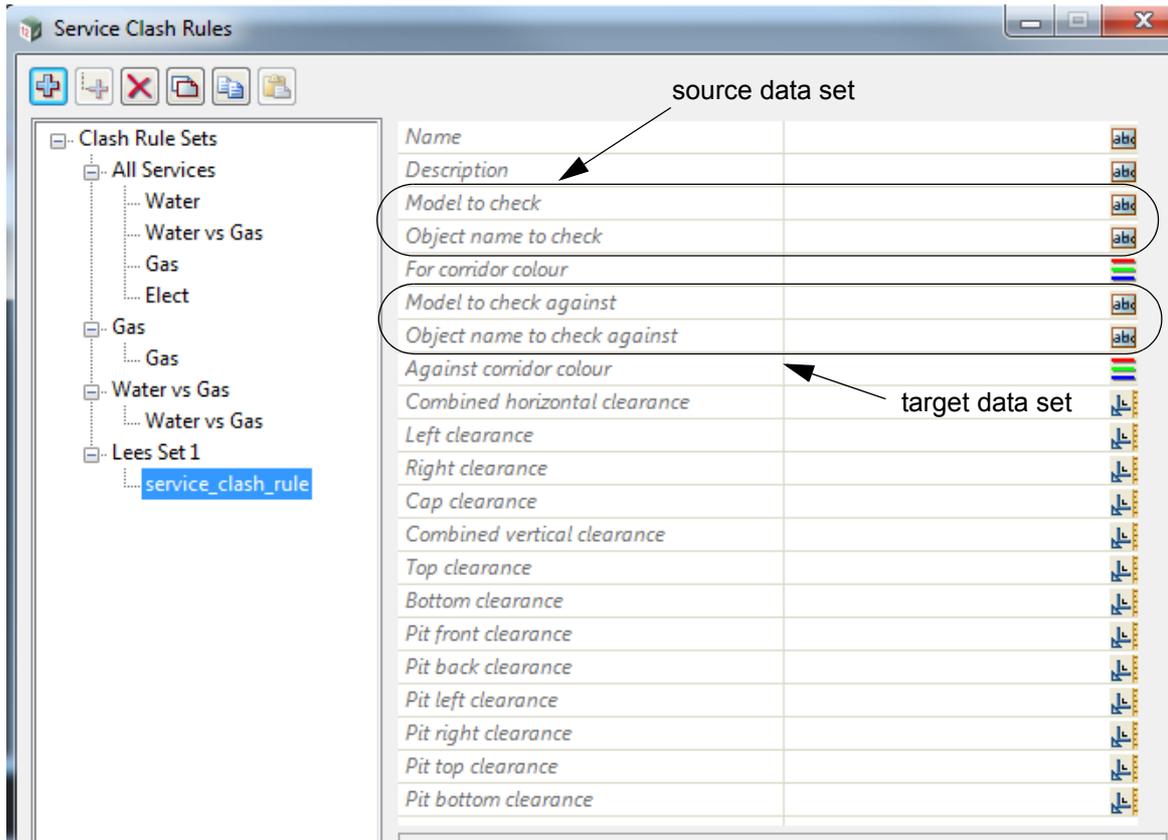


(b) Rules in a Clash Rule Set

The rules for a named Clash Rule Set as defined as second level items under the *Clash Rule Set*. The rules define what is meant by a particular clash.

A new *Rule* is created by clicking and highlighting the named *Clash Rules Set* and then pressing the **Add Child** button, or by clicking on and highlighting a *Rule* already in the named *Clash Rule Set* and then pressing the **Add** button.

A new *Rule* with the dummy name **service\_clash\_rule** is then created and a new name must be entered in the **Name** field on the right hand side of the tree. The **Description** is optional.



The fields and buttons used in the right hand side of the panel for a **Clash Rule** has the following functions.

Field	Description	Type	Defaults	Pop-Up
<b>Name</b>	<i>name of the rule. This must be unique within this set of rules.</i>	text box		
<b>Description</b>	<i>description of the this rule.</i>	text box		
<b>Models to check</b>	<i>names of models to check for clashes against the models to check against. This can include wild cards (*) and wild characters (?).</i>	text box		
<b>Object names to check</b>	<i>if not blank, the names of the objects to check for clashes is restricted to these names from the models selected with <b>Models to check</b>. This can include wild cards (*) and wild characters (?).  <i>If blank then all objects in the models are checked for clashes.</i></i>	text box		
<b>For corridor colour</b>	<i>colour of the defined corridor around the objects being checked for clashes. The corridors around the objects are only created if the <b>Model for check corridors</b> is non blank in the <b>Service Clash Detection</b> panel (see <a href="#">22.8.1 Clash Detection</a>).</i>	colour box		available colours
<b>Models to check against</b>	<i>names of models to check the <b>Models to check</b> against. This can include wild cards (*) and wild characters (?).</i>	text box		

**Object names to check against** text box

*if not blank, the names of the objects to check against for clashes is restricted to these names from the models selected with **Models to check against**. This can include wild cards (\*) and wild characters (?).*

*If blank then all objects in the models are checked against for clashes.*

**Against corridor colour** colour box available colours

*colour of the defined corridor around the objects being checked against for clashes. The corridors around the objects are only created if the **Model for check against corridors** is non blank in the **Service Clash Detection** panel (see [22.8.1 Clash Detection](#).)*

**Combined horizontal clearance** real box

*if non blank, the total clearance for both the left and right of round pipes or culverts.*

*If the sum of the **Left clearance** and **Right clearance** is less than the **Combined horizontal clearance**, then half of the difference between the **Combined horizontal clearance** and the sum of the **Left and Right clearances** is added to the **Left clearance** and half to the **Right clearance**.*

**Left clearance** real box

*if non blank, the minimum clearance required on the left of round pipes or culverts.*

**Right clearance** real box

*if non blank, the minimum clearance required on the right of round pipes or culverts.*

**Cap clearance** real box

*if non blank, the clearance required at the end of each round pipes or culvert.*

**Combined vertical clearance** real box

*if non blank, the total clearance for both top and bottom of round pipes or culverts.*

*If the sum of the **Top clearance** and **Bottom clearance** is less than the **Combined vertical clearance**, then half of the difference between the **Combined vertical clearance** and the sum of the **Top and Bottom clearances** is added to the **Top clearance** and half to the **Bottom clearance**.*

**Top clearance** real box

*if non blank, the minimum clearance required on the top of round pipes or culverts.*

**Bottom clearance** real box

*if non blank, the minimum clearance required on the bottom of round pipes or culverts.*

**Pit front clearance** real box

*if non blank, the clearance required on the front of drainage and sewer pits.*

**Pit back clearance** real box

*if non blank, the clearance required at the back of drainage and sewer pits.*

**Pit left clearance** real box

*if non blank, the clearance required on the left of drainage and sewer pits.*

**Pit right clearance** real box

*if non blank, the clearance required on the right of drainage and sewer pits.*

**Pit top clearance** real box

*if non blank, the clearance required on the top of drainage and sewer pits.*

**Pit bottom clearance** real box

*if non blank, the clearance required at the bottom of drainage and sewer pits.*

**Clash Rules and Datasets Explained:**

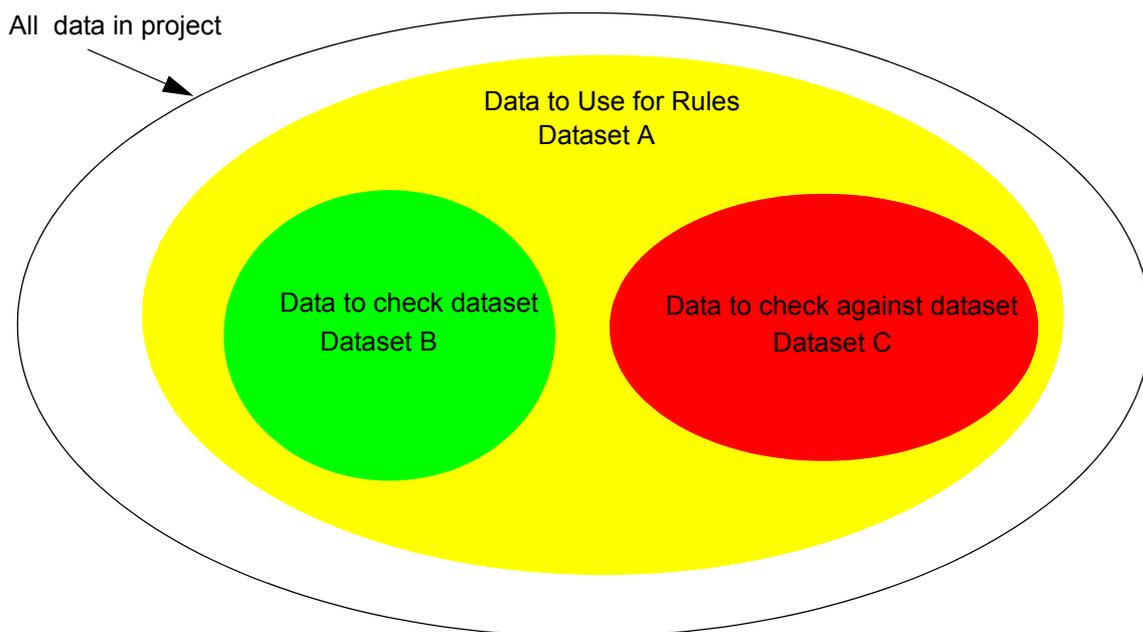
1. There must be at least one clash rule in a Clash Rule Set otherwise there is nothing to do. If a Clash Rule Set has no rules then no results will be reported.
2. A Rule Set can contain more than one rule. In such a case when the Clash Detection is run with the Rule Set, each rule will be processed separately and reported to the one file BUT each set will be reported in a different section of the file (unless the report template has been customised).
3. When the **Service Clash Detection** panel is run, a dataset of objects is selected by the user using the **Data to use for rules** source Box. This can be called **Dataset A**. Typically, this would contain all the modelled utilities and objects that you want to check for clashes in this run.

Each Rule defines two separate datasets - the **Data to check dataset (Dataset B)** containing the elements being checked, and the **Data to check against dataset (Dataset C)** containing the elements being checked against.

The **Data to check dataset (Dataset B)** is taken from the **Overall Data to Use (Dataset A)**.

The **Data to check against dataset (Dataset C)** is taken from the **Overall Data to Use (Dataset A)** minus the **Data to check dataset (Dataset B)**.

So elements **cannot** exist in both the **Data to check dataset** and the **Data to check against dataset**. They are in one or the other, or neither.



4. In order to select which elements are in the **Data to check dataset**, each rule has two filter fields: one for the models (**Models to check**) and another for the names of objects in those models (**Object names to check**).  
The **Models to check** field can contain wildcards (\*) and wild characters (?) or if the field is blank, it will match all models in the Overall data to use. That is, a blank field is interpreted as the wildcard \* which selects all models.  
The **Object names to check** field can contain wildcards (\*) and wild characters (?) or if the field is blank, it will match all objects in the **Models to check**. That is, a blank field is interpreted as the wild character \* which selects all elements in the **Models to check**.
5. In order to select which elements are in the **Data to check against dataset**, each rule has two filter fields: one for the models (**Models to check against**) and another for the names of objects in those models (**Object names to check against**).

The **Models to check against** field can contain wildcards (\*) and wild characters (?) or if the field is blank, it will match all models in the Overall data to use. That is, a blank field is interpreted as the wild card \* which selects all models.

The **Object names to check against** field can contain wildcards (\*) and wild characters (?) or if the field is blank, it will match all objects in the **Models to check against**. That is, a blank field is interpreted as the wild card \* which selects all elements in the **Models to check against**.

6. For example:

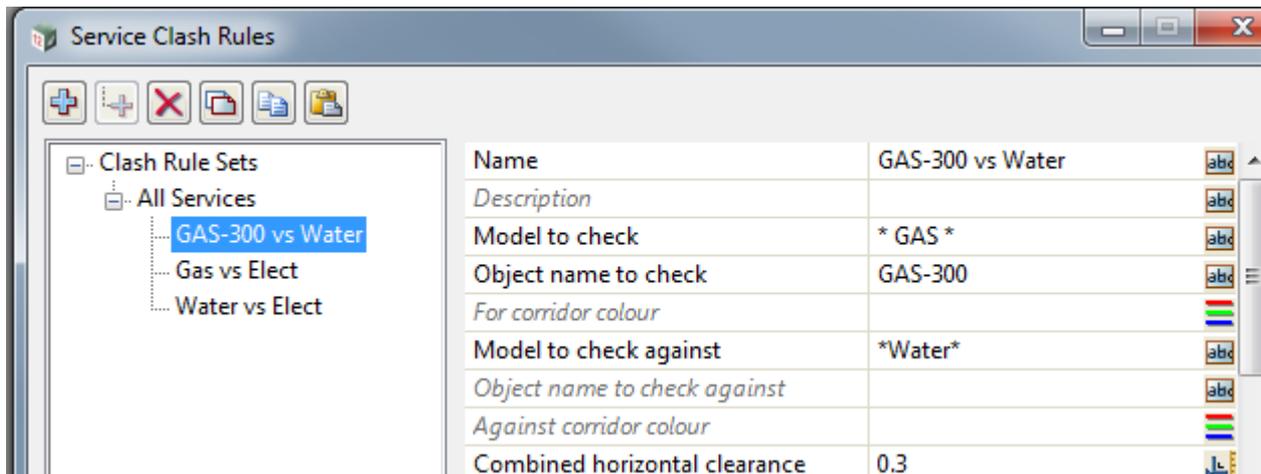
**Models in the Data Source "Data to use for Rules"**

ABANDONED GAS, ELECTRICITY EXISTING, ELECTRICITY PROPOSED  
 GAS EXISTING, GAS PROPOSED, SEWER,  
 STORMWATER, WATER PROPOSED

**Some strings in GAS EXISTING and GAS PROPOSED models**

GAS-300, GAS-150

**Clash Rule GAS-300 vs Water**



**Models to Check Filter**

\* GAS \*

**Matching Models**

GAS EXISTING, GAS PROPOSED, ABANDONED GAS

**Object Names to Check Filter**

\*300

**Matching strings**

GAS-300

**Models to Check against Filter**

\*Water\*

**Matching Models**

STORMWATER, WATER PROPOSED

**Object names to check against Filter**

blank

**Matching Strings**

any strings in the models STORMWATER and WATER PROPOSED

So the strings GAS-300 in the models GAS EXISTING, GAS PROPOSED and ABANDONED GAS are checked against any strings in the models STORMWATER and WATER PROPOSED.

7. Elements in the **Data to check dataset** will **only be checked** against elements in the **Data to check against dataset**. and any other objects not in these datasets are ignored

So you should be specific for the **Data to check dataset** and not too generic, since being too

generic will mean that there are few elements left to be in the **Data to check against dataset** leaving little or nothing to check against.

Hence you should be fairly specific in your **Data to check dataset** filters, but can be totally generic in the **Data to check against dataset** filters.

**Recommendations for Using the Service Clash Detection:**

(a) PLAN AHEAD

Think about what clashes you are likely to need to check for. Try to consider all possibilities,

(b) Model each type of utility in its own model, e.g. GAS, WATER, ELEC, TELECOM.

Do not mix utilities of different types in the same model.

(c) As a minimum create a Rule for each service type.

These can still be underneath a single Rule Set.

(d) If there are different clearances for different services being checked, create a rule for each case. e.g. Gas vs Water, Gas vs Elec, Gas vs Other.

(e) If you need to check for clashes between utilities of the same type, place these in different models or name them differently so that they can be selected by name, e.g. WATER TRUNK and WATER CONNECTIONS or ELEC LEFT and ELEC RIGHT.

(f) If there are different clearances for different diameters/sizes, ensure the diameter/size is contained in either the model name or preferably in the string name. For example GAS-PROP-300.

This allows them to be more easily selected with the filters.

(g) Be as narrow and specific with the **Data to check dataset** filters as you can be.

There is a trade-off, however, between being specific and needing to create multiple rules. Find a balance.

(h) Use standard naming for models and strings and develop standard clash detection rule sets.

(i) You can use map files to help filter out, or map strings to certain models, to match existing rule sets.

## 23. MTF Edit

The **MTF Edit** menu and all the options on it, especially *Modify left* and *Modify right*, have undergone a combined heart, lung and brain transplant, along with extensive cosmetic surgery.

**Smart Chainages** have been made even smarter and used throughout all the *MTF Edit* options making "chainage free design" even easier.

The new concepts of **Layers** and **Shapes** along with the new **Fixed =>Decisions** and **Fixed =>Boxing** means that in many situations the *MTF Left/Right Modifier* commands can now replace **Decisions** in **Templates** and the need for separate **Boxing**.

These new tools can also generate **trimeshes** for representing pavement layers and gutters as 3D objects.



A major change for V10 MTF projects is that the following **MTF Modify Left/Right Create** commands no longer exist

*Fixed =>Width, Fixed =>Height, Fixed =>Xfall, Fixed =>Xfall CRC*

*Cut =>Width, Cut =>Height, Cut =>Slope*

*Fill =>Width, Fill =>Height, Fill =>Slope*

have been replaced by

***Fixed =>Link, Cut =>Link, Fill =>Link***

And

*Fixed =>from link =>Width from link, Fixed =>from link=> Height from, Fixed =>from link=> Xfall from link*

*Cut =>from link =>Width from link, Cut =>from Link=> Height from link, Cut =>from link=> Slope from link*

*Fill =>from link =>Width from link, Fill =>from Link=> Height from, Fill =>from link=> Slope from link*

have been replaced by

***Fixed =>from link, Cut =>from link, Fill =>from link***

And similarly for the **to string**, **to tin**, **to RL** and **to 2 strings** options.

So you need to carefully read the section [23.1 Migrating MTFs from V10 to V11](#)

See

[23.1 Migrating MTFs from V10 to V11](#)

[23.2 Smart Chainages in MTF Edit](#)

[23.3 MTF Edit - Reorganised](#)

[23.4 MTF Edit =>Hinge](#)

[23.5 MTF Edit =>Modify Left/Right](#)

[23.6 MTF Edit =>Boxing](#)

[23.7 MTF Edit =>More](#)

[23.8 MTF Edit =>Settings](#)

[23.9 MTF Edit =>Save](#)

## 23.1 Migrating MTFs from V10 to V11

See

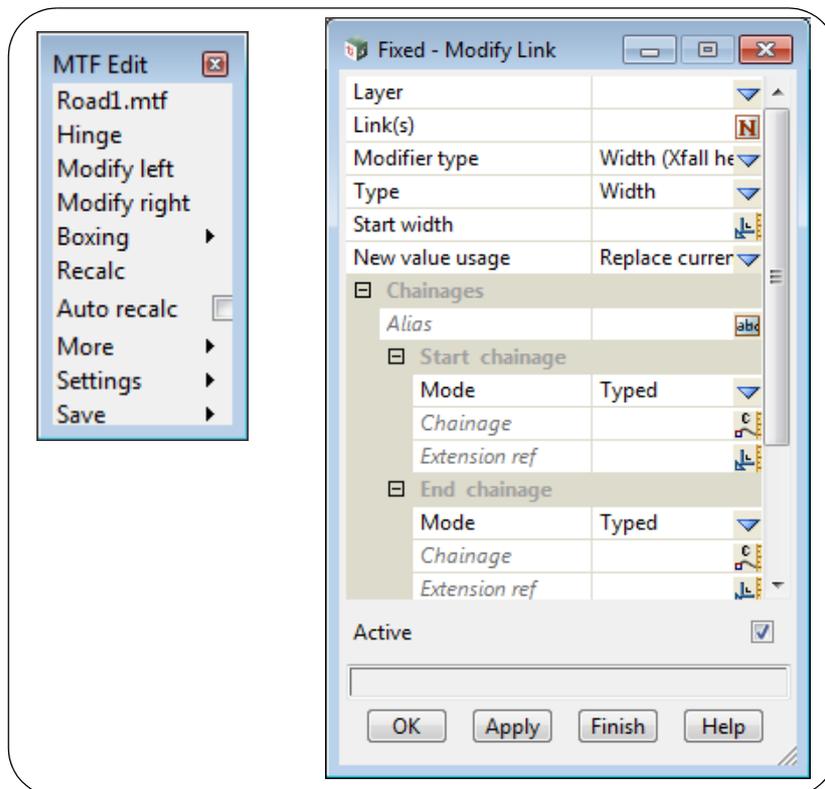
[23.1.1 New Look for MTF Edit and Left/Right Modifiers](#)

[23.1.2 New MTF Modify Left/Right Link Commands](#)

[23.1.3 Automatic Conversion of V10 MTFs to V11 MTFs](#)

### 23.1.1 New Look for MTF Edit and Left/Right Modifiers

The **MTF Edit** menu has been completely rearranged and the **Left/Right Modifier** command panels have a totally new look.



There are new fields **Layer** and **Alias**, and the **Absolute** tick box has been replaced by the choice box **New value usage**.

**Layer** is defaulted to the value **Design** so does not have to be changed for things to work as they did in V10.

**Alias** is by default **blank** and does not have to be changed for things to work as they did in V10.

**Absolute** ticked **on** is converted to **New value usage** with the choice **Replace the current value of the link**.

**Absolute** ticked **off** is converted to **New value usage** with the choice **Add to the current value of the link**.

Also be aware that in V11 it is possible to turn off displaying the **Extra start** and **Extra end** Tick boxes on the panels and the **Extra start** and **Extra end** column the grid on the **Left/Right Modifiers** panel.

## 23.1.2 New MTF Modify Left/Right Link Commands

A major change in the V11 MTF commands is that the following **MTF Modify Left/Right Create** commands **no longer exist**:

*Fixed =>Width, Fixed =>Height, Fixed =>Xfall, Fixed =>Xfall CRC*

*Cut =>Width, Cut =>Height, Cut =>Slope*

*Fill =>Width, Fill =>Height, Fill =>Slope*

and have been replaced by

***Fixed =>Link, Cut =>Link, Fill =>Link***

And

*Fixed =>from link =>Width from link, Fixed =>from link=> Height from, Fixed =>from link=> Xfall from link*

*Cut =>from link =>Width from link, Cut =>from Link=> Height from link, Cut =>from link=> Slope from link*

*Fill =>from link =>Width from link, Fill =>from Link=> Height from, Fill =>from link=> Slope from link*

have been replaced by

***Fixed =>from link, Cut =>from link, Fill =>from link***

And similarly for the **to string**, **to tin**, **to RL** and to **2 strings** options.

The new modify **Link** commands cover **all** the cases of the old commands, plus allow many more cases.

The first major advantage of the new modify **Link** command is that you no longer have to know how a link was originally created (e.g. width and xfall, or width and height etc).

Regardless of how the link was created, any one of width, height, xfall or slope can be modified by the **Link** command.

### 23.1.3 Automatic Conversion of V10 MTFs to V11 MTFs

When a *V10 MTF* is first used in V11, wherever possible a V10 MTF command is migrated to the equivalent V11 MTF command with **Layer** set to **Design**, **Alias** left blank and **Absolute** converted to **New value usage**.

However the superseded commands

*Fixed =>Width, Fixed =>Height, Fixed =>Xfall, Fixed =>Xfall CRC*

*Cut =>Width, Cut =>Height, Cut =>Slope*

*Fill =>Width, Fill =>Height, Fill =>Slope*

did not have enough information in them to know exactly which **Modifier type** to map them to in the *Fixed =>Link, Cut =>Link, Fill =>Link* commands.

Similarly for the other superseded **from tin, to string, to tin, to RL** and to **2 strings** commands.

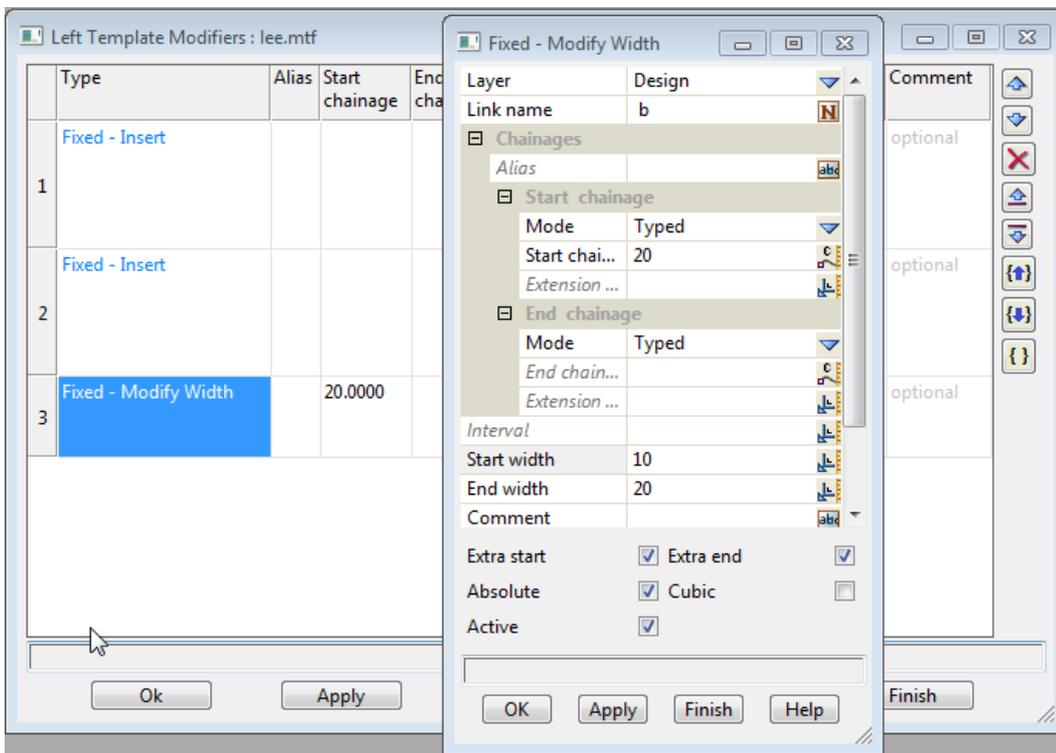
So in V11, the superseded commands will still exist and the V10 commands will be migrated to the same command in V11.

**HOWEVER**, although in V11 you will be able to click on and edit these superseded commands if they already exist in the MTF, you will not be able to create new ones.

For example, the **Fixed =>Width** command in a **Left Modifiers** panel will be converted to a V11 **Fixed - Modify Width** row in the V11 **Left Modifiers** panel.

And if in V11 you then click on the **Fixed - Modify Width** row, a **V11 Fixed - Modify Width** panel will be displayed and allow you make changes to the **Fixed - Modify Width** row.

But the **Fixed =>Width** command will not appear on the V11 **MTF Edit** menu for you to create a new one.

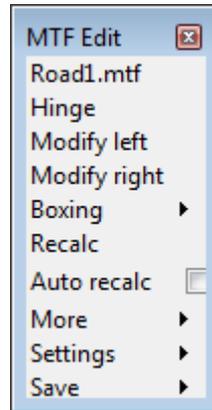


## 23.2 Smart Chainages in MTF Edit

For the new information on **Smart Chainage**, see [9. Smart Chainages](#).

## 23.3 MTF Edit - Reorganised

The **MTF Edit** menu has been completely rearranged.



None of the V10 options have been removed but they have been moved around so the menu is smaller and the more common options are on the top level.

**Modify left** and **Modify right** were the most used options so they are now on the **MTF Edit** menu itself so that you no longer have to walk right each time you need them.

See

[23.4 MTF Edit =>Hinge](#)

[23.5 MTF Edit =>Modify Left/Right](#)

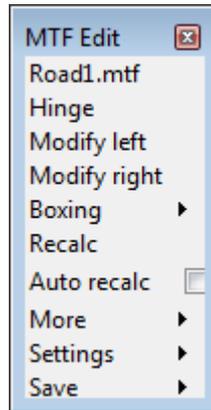
[23.6 MTF Edit =>Boxing](#)

[23.7 MTF Edit =>More](#)

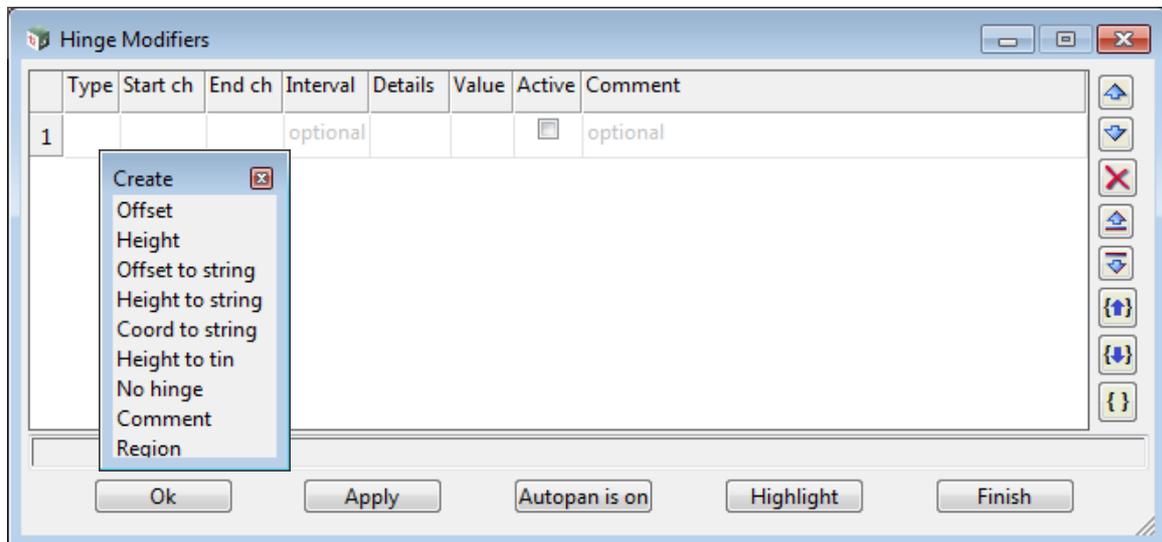
[23.8 MTF Edit =>Settings](#)

[23.9 MTF Edit =>Save](#)

## 23.4 MTF Edit =>Hinge



Clicking on **Hinge** brings up the **Hinge Modifiers** panel.



See

- [23.4.1 Tick Boxes in the Grid](#)
- [23.4.3 Env Variable to Turn Off Interval Column](#)
- [23.4.4 Active Column in Hinge Grid](#)
- [23.4.5 New Look for Hinge Modifier Panels](#)
- [23.4.6 Autopan Button](#)
- [23.4.7 Regions](#)
- [23.4.8 Height To Tin](#)
- [23.4.9 Smart Chainages for Hinge Modifiers](#)

## 23.4.1 Tick Boxes in the Grid

The Grid had been upgraded and there are now tick boxes instead of **Yes/No** choices in the **Extra start**, **Extra end** and **Active** columns.

## 23.4.2 Turning Off Extra Start/End

In V11 it is possible to not display the **Extra start** and **Extra end** tick boxes on the **MTF Hinge Command** panels and not display the **Extra start** and **Extra end** columns in the **Hinge Modifiers** panel.

This is controlled by the MTF Setting **Show Extra Start/End?**. See [23.8.5 MTF Edit =>Settings =>Show Extra Start/end?](#).

## 23.4.3 Env Variable to Turn Off Interval Column

There is an environment variable

NEW\_MTF\_EDITOR\_SHOW\_INTERVAL\_COLUMN\_4D

and when it has value zero, the **Interval** column is not displayed in the Grid.

In the **Edit Environment Variables** panel brought up by the option

**Project =>Management =>env.4d**

the tick box of the env variable is on the page

Env.4d >MTF & Boxing >MTF Editor General

and is called **Editor, show interval column**.

## 23.4.4 Active Column in Hinge Grid

There is now an **optional Active** column in the **Hinge Modifiers** grid so that the individual lines of the grid can be turned off and so not used in the **Hinge Modifiers**. This replaces having to delete and reinsert the line.

If the tick box in the **Active** column is **ticked**, then the line is **used**.

If the tick box in the **Active** column is **not ticked**, then the line is **not used**.

## 23.4.5 New Look for Hinge Modifier Panels

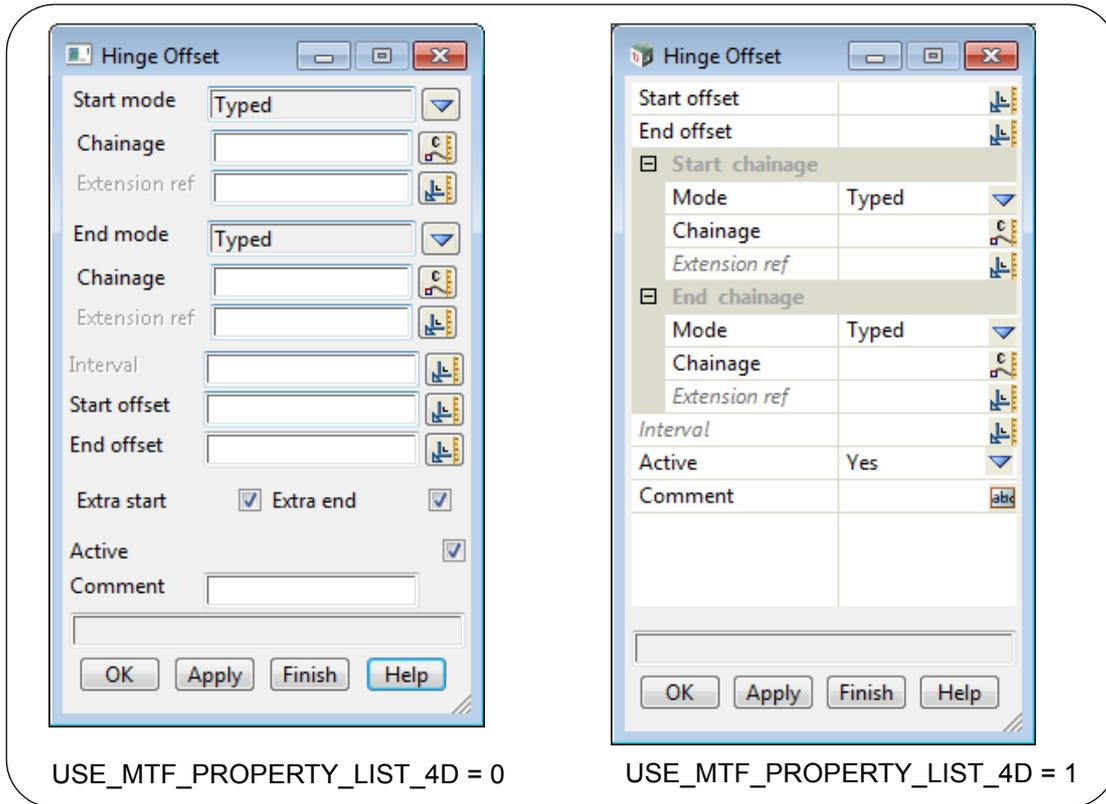
In V11 the panels for the **Hinge Modifiers** have been changed to use what is know as Property Lists and look different to V10.

The new look is on by default but can be controlled by the environment variable

USE\_MTF\_PROPERTY\_LIST\_4D

If USE\_MTF\_PROPERTY\_LIST\_4D = 1, the new property lists are used in the Hinge Modifier panels.

If USE\_MTF\_PROPERTY\_LIST\_4D = 0, the new property lists are **not** used in the Hinge Modifier panels and the panels will have the same look as in V10.



**Note:**

There is no special field for this env variable in the **Edit Environment Variables** panel so to set the value to 0 instead of using the default value of 1, you need to add the environment variable **USE\_MTF\_PROPERTY\_LIST\_4D** and its value 0 to the **Variables** section of the **Edit Environment Variables** panel.

### 23.4.6 Autopan Button

There is now an **Autopan** button and when it is clicked to say **Autopan is on** then if the chainage range is not in the view that the Reference string is on then the view will be panned so that it is on the view.

### 23.4.7 Regions

**Regions** are now available as a command in the **Hinge Modifiers** grid.

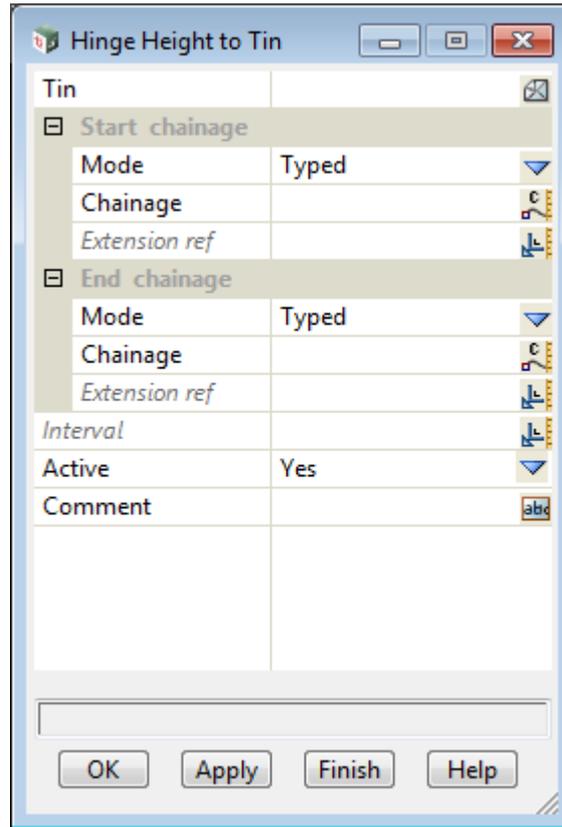
There are **Regions**, **Previous Region** and **Next Region** icons at the right hand end of the **Hinge Modifiers** panel.

See [12.11 Regions in Some Grids](#).

### 23.4.8 Height To Tin

There is a new **Hinge to Tin** command that gives the Hinge string the height at the tin.

Selecting **Height to tin** brings up the **Hinge Height to Tin** panel



The fields and buttons used in this panel have the following functions.

Field	Description	Type	Defaults	Pop-Up
Tin		tin box		

*tin to use for z-values. At each vertex of the hinge string in the chainage range, set the z-value to be the tin z-value at the same (x,y) position as the vertex.*

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

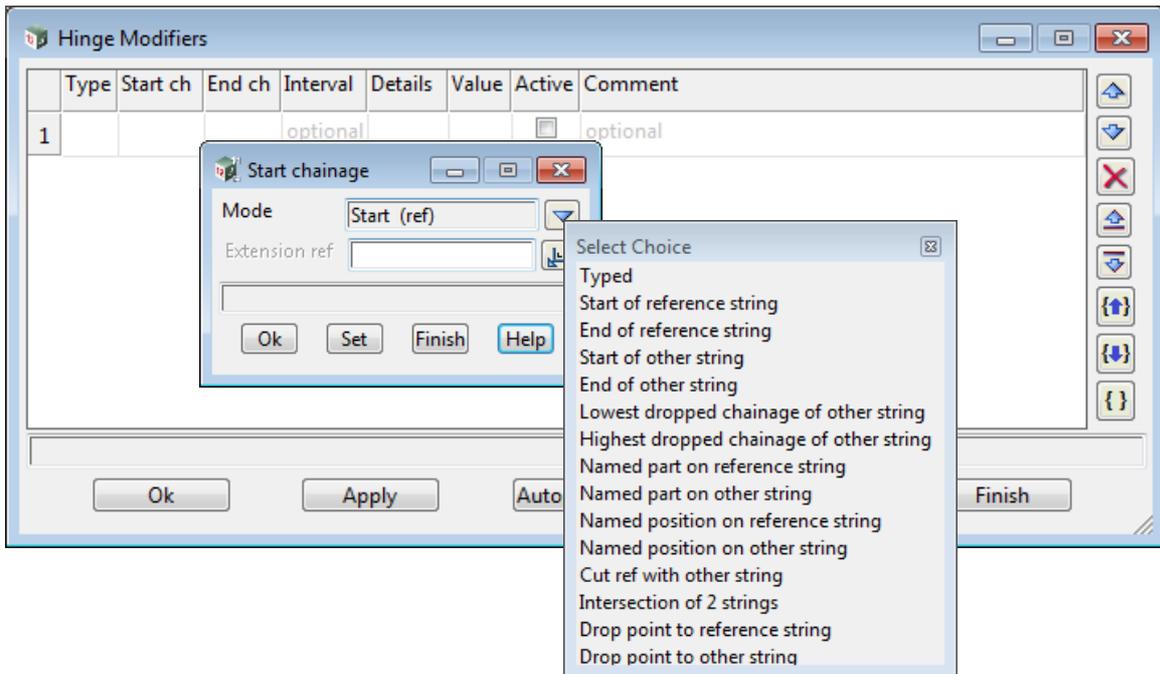
*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 23.4.9 Smart Chainages for Hinge Modifiers

**Smart Chainages** are now being used in the grid in the **Hinge Modifiers** panel that is brought up by the option

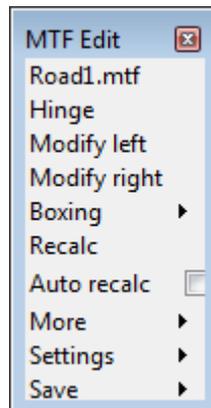
**MTF Edit =>Hinge**

If you right click in the **Start Chainage** or **End Chainage** column or select **Browse** if the *Browse* menu comes up, then a **Start Chainage** or **End Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**. See [23.3 MTF Edit - Reorganised](#).



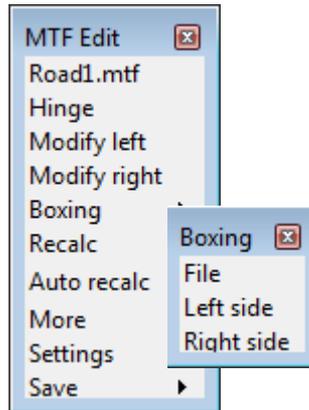
There is also an **Start/End Extension** field in the **Start/End Chainage** panel for adding to the reference chainage calculated by the **Smart Chainage**.

## 23.5 MTF Edit =>Modify Left/Right



There has been major changes to the Modify Left/Right options so they are in their own Chapter [24. MTF Edit =>Modify Left/Right](#).

## 23.6 MTF Edit =>Boxing

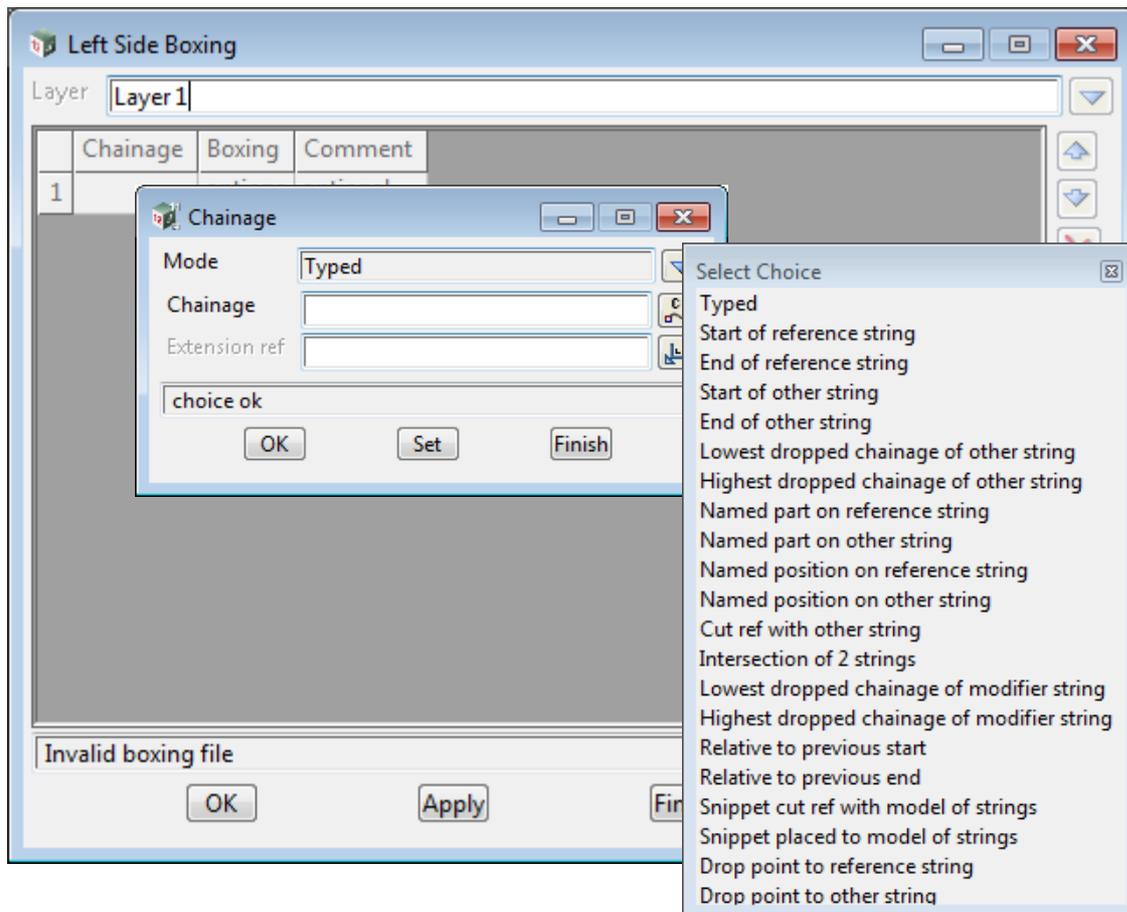


**Smart Chainages** are now being used in the grids for the **Left Side Boxing** and **Right Side Boxing** panels that are brought up by the options

**MTF Edit =>Boxing =>Left side**

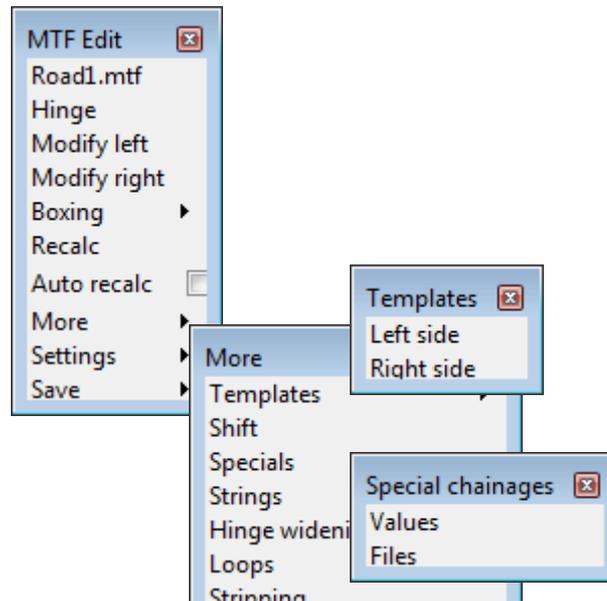
**MTF Edit =>Boxing =>Right side**

If you right click in the **Chainage** column or select **Browse** if the Browse menu comes up then a **Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**. See [23.3 MTF Edit - Reorganised](#).



There is also an **Extension** field in the **Chainage** panel for adding to the chainage calculated by the **Smart Chainage**.

## 23.7 MTF Edit =>More



Apart from being in a new menu position, the options under **More** work the same as they did in **12d Model 10** except for the additions of Smart Chainages in some Grids.

See

[23.7.1 Smart Chainages in MTF Templates Grid](#)

[23.7.2 Smart Chainages in MTF Shift](#)

[23.7.3 Smart Chainages in MTF Specials Values Grid](#)

[23.7.4 Smart Chainages in MTF String Modifiers Grid](#)

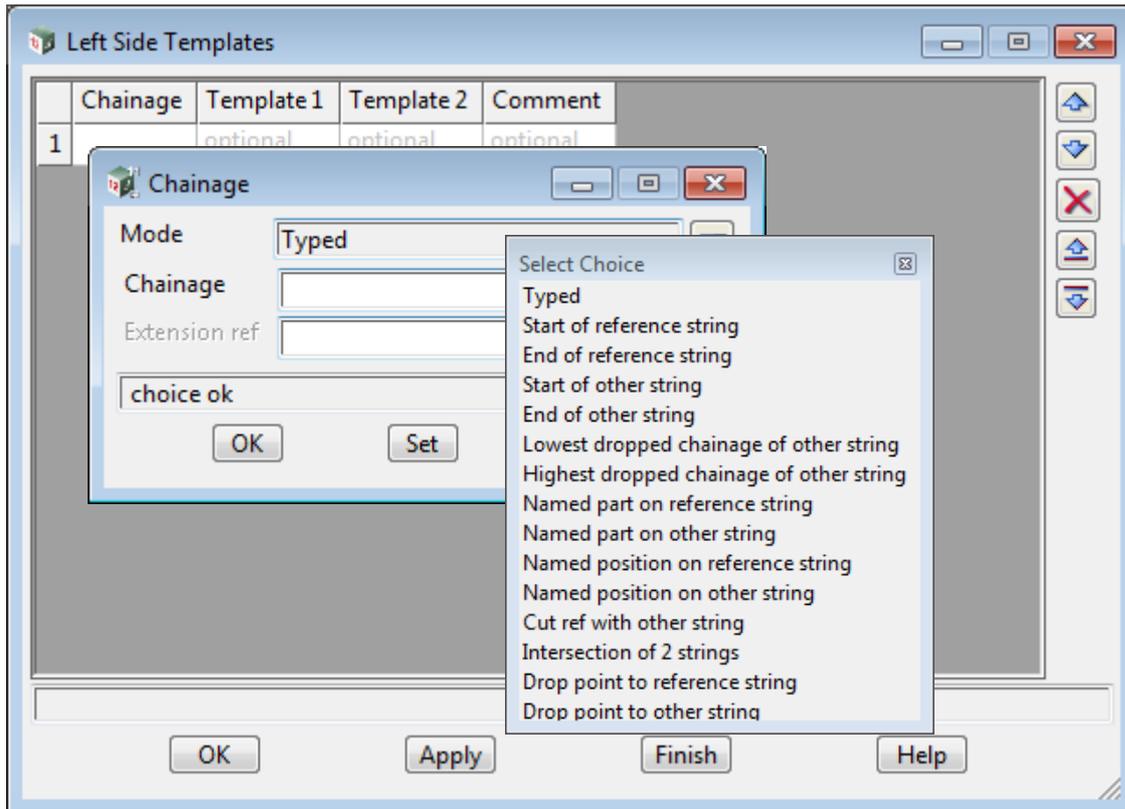
## 23.7.1 Smart Chainages in MTF Templates Grid

**Smart Chainages** are now being used in the grids for the **Left Side Templates** and **Right Side Templates** panels that are brought up by the options

**MTF Edit =>More =>Templates =>Left side**

**MTF Edit =>More =>Templates =>Right side**

If you right click in the **Chainage** column or select **Browse** if the *Browse* menu comes up, then a **Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**. See [23.3 MTF Edit - Reorganised](#).



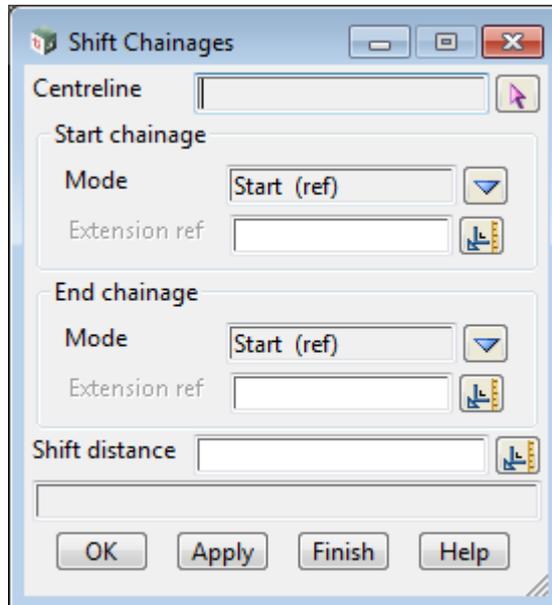
There is also an **Extension** field in the **Chainage** panel for adding to the reference chainage calculated by the **Smart Chainage**.

## 23.7.2 Smart Chainages in MTF Shift

**Smart Chainages** are now used for **Chainage to shift from** and **Chainage to shift to** in the option **MTF Edit =>More =>Shift**

The **Reference string** may have to be selected to enable the Smart chainages.

**Note** that the **Shift distance** will only be applied to **Start** and **End chainages** with **Mode Typed**.

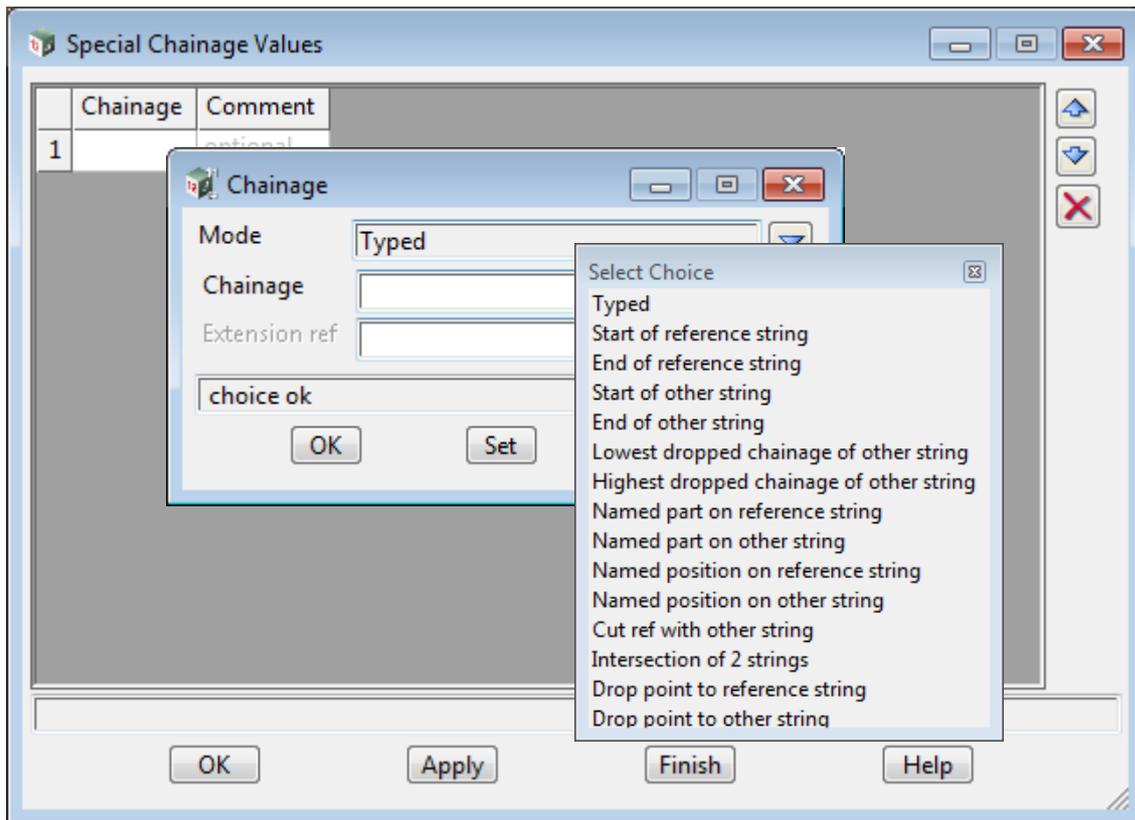


### 23.7.3 Smart Chainages in MTF Specials Values Grid

**Smart Chainages** are now being used in the grid for the **Special Chainage Values** panel that is brought up by the option

**MTF Edit =>More =>Specials =>Values**

If you right click in the **Chainage** column or select **Browse** if the *Browse* menu comes up, then a **Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**. See [23.3 MTF Edit - Reorganised](#).



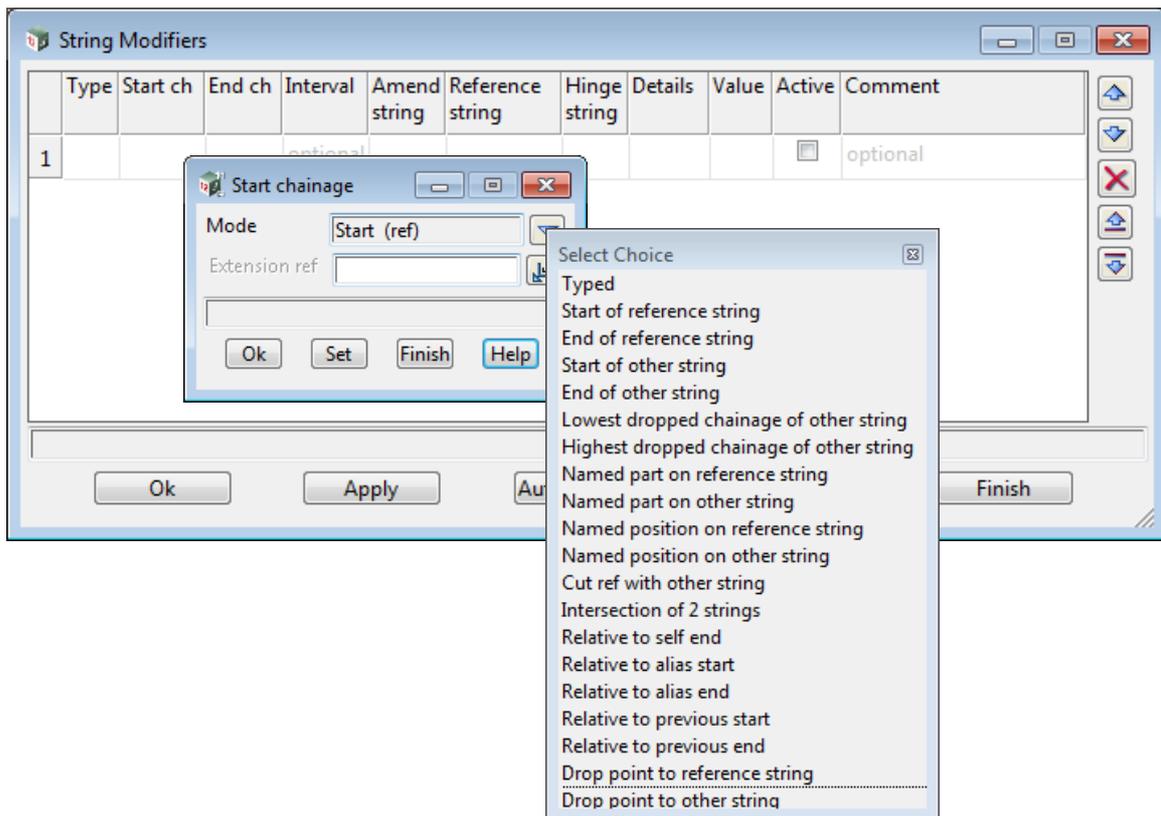
There is also an **Extension** field in the **Chainage** panel for adding to the reference chainage calculated by the **Smart Chainage**.

## 23.7.4 Smart Chainages in MTF String Modifiers Grid

**Smart Chainages** are now being used in the grid in the **String Modifiers** panel that is brought up by the option

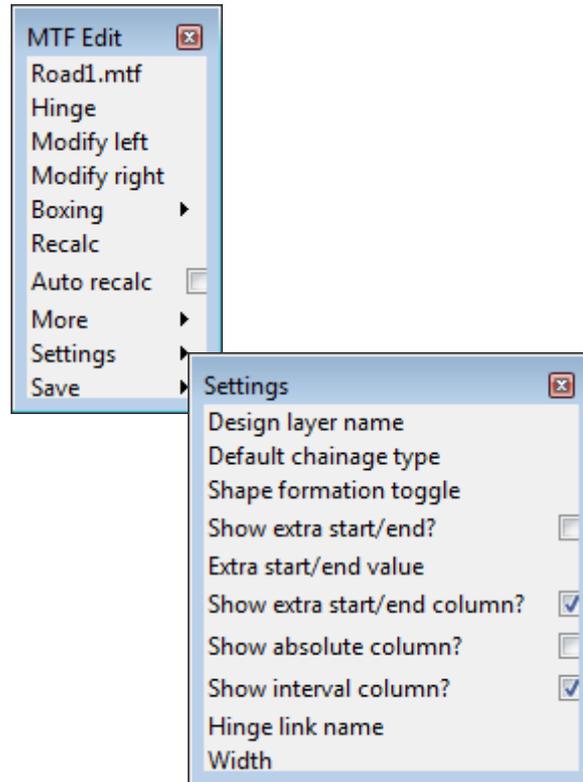
**MTF Edit =>More =>Strings**

If you right click in the **Start Chainage** or **End Chainage** column or select **Browse** if the *Browse* menu comes up, then a **Start Chainage** or **End Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**. See [23.3 MTF Edit - Reorganised](#).



There is also an **Start/End Extension** field in the **Start/End Chainage** panel for adding to the reference chainage calculated by the **Smart Chainage**.

## 23.8 MTF Edit =>Settings



Apart from being in a new menu position, **Auto recalc**, **Loops**, **Super/Widening** and **Width** work the same as they did in **12d Model 10**.

**Stripping** now has smart chainages in its grid and for this and the new **MTF Settings**, see

[23.8.1 Smart Chainages in MTF Stripping Grid](#)

[23.8.2 MTF Edit =>Settings =>Design layer name](#)

[23.8.3 MTF Edit =>Settings =>Default chainage type](#)

[23.8.4 MTF Edit =>Settings =>Shape formation toggle](#)

[23.8.5 MTF Edit =>Settings =>Show Extra Start/end?](#)

[23.8.6 MTF Edit =>Settings =>Extra Start/end value](#)

[23.8.7 MTF Edit =>Settings =>Show extra start/end column?](#)

[23.8.8 MTF Edit =>Settings =>Show absolute column?](#)

[23.8.9 MTF Edit =>Settings =>Show interval column?](#)

[23.8.10 MTF Edit =>Settings =>Hinge link name](#)

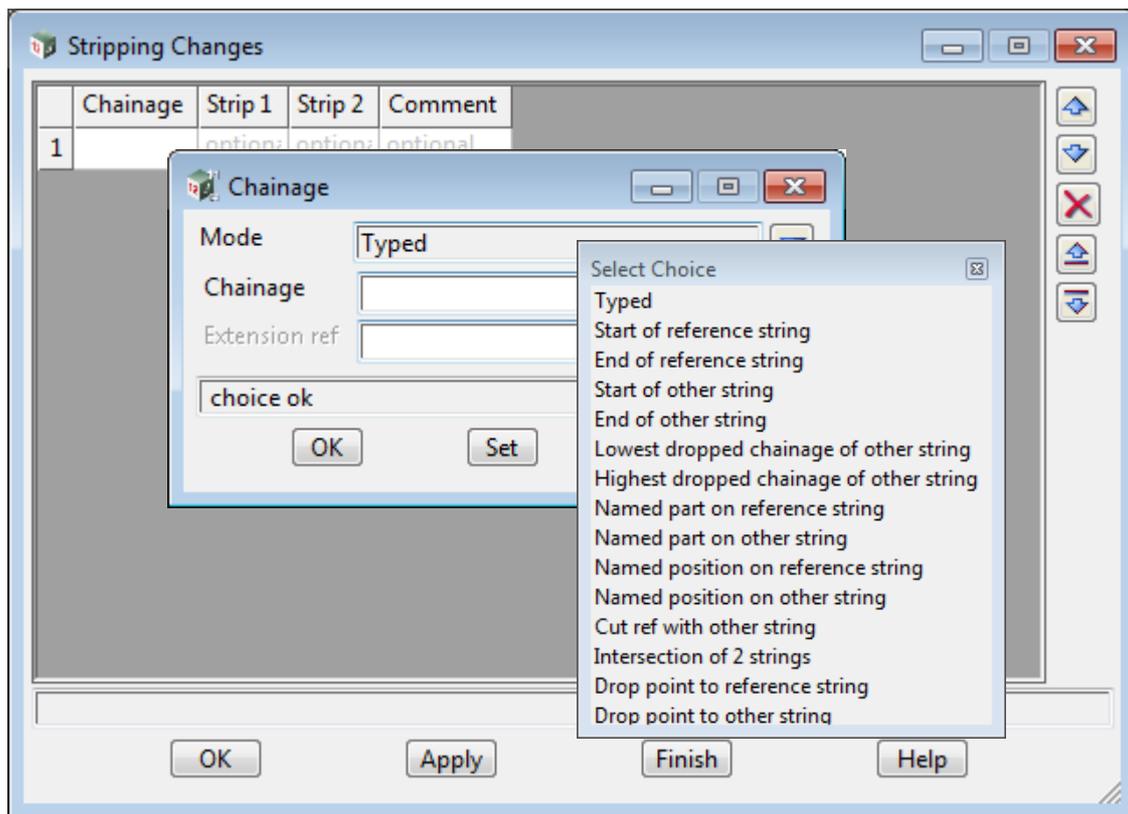
[23.8.11 MTF Edit =>Settings =>Hinge widening treatment](#)

## 23.8.1 Smart Chainages in MTF Stripping Grid

**Smart Chainages** are now being used in the grid for the **Stripping Changes** panel that is brought up by the option

**MTF Edit =>Settings =>More =>Stripping**

If you right click in the **Chainage** column or select **Browse** if the *Browse* menu comes up, then a **Chainage** panel comes up which has the most of the standard MTF choices for **Smart Chainages**. See [23.3 MTF Edit - Reorganised](#).



There is also an **Extension** field in the **Chainage** panel for adding to the reference chainage calculated by the **Smart Chainage**.

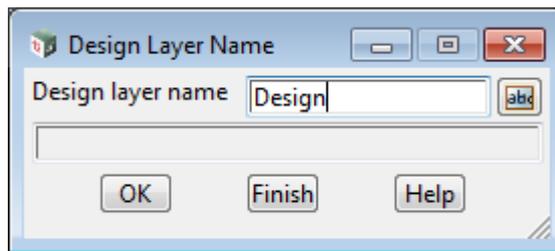
## 23.8.2 MTF Edit =>Settings =>Design layer name

In **12d Model 10**, the *MTF Modifiers* only produced the one layer and it was called **Design**.

In **12d Model 11**, the *MTF Modifiers* can produce any number of named layers but there is one special layer referred to as the **Design Layer**, that is used to produce design x sections, design strings, the road tin etc.

Any MTF points in the Design Layer will automatically generate points on the design x sections, and design strings will be generated without them having to be done by the **Create =>Strings** MTF Modifier command.

The name of the special Design Layer can be changed by clicking on **Design layer name** to bring up the **Design Layer Name** panel.



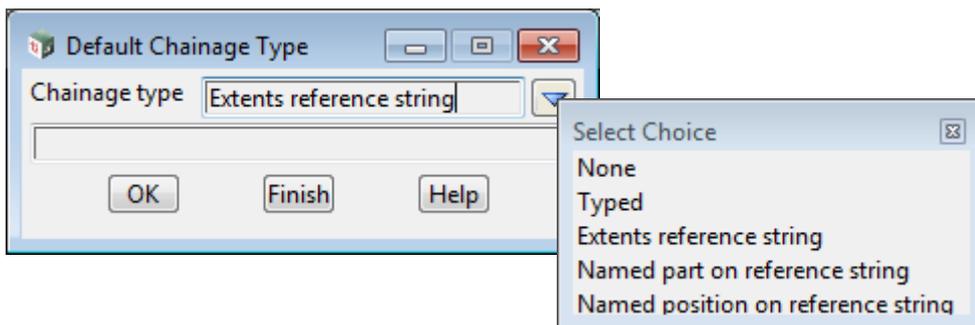
**Design layer name** - the name used for the special Design Layer.

**OK** - selecting *OK* sets the default name for the design layer.

The default name is **Design** so that if you are not using any of the additional layers available in **12d Model 11**, things work the same as they did in **12d Model 10** with just the one layer that has the name **Design**.

## 23.8.3 MTF Edit =>Settings =>Default chainage type

The **Default Chainage Type** panel specifies the default Smart Chainage Mode to use in the MTF Modifier commands in this MTF file.



The choices are:

**None** - no Smart Chainage choice is used as the default.

**Typed** - *Typed* is the default for Smart Chainages.

**Extents reference string** - the start of the reference string is used for *Start modes* and the end of the reference string is used for *End modes*.

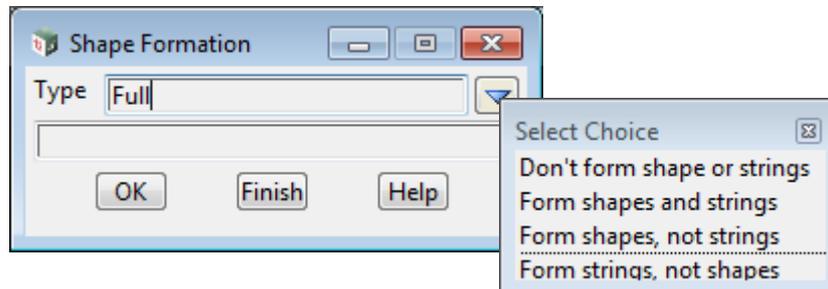
**Named part on reference string** - *named part on reference string* is the default for Smart Chainages.

**Named position on reference string** - *named position on reference string* is the default for Smart Chainages.

**OK** - selecting *OK* sets the given *Chainage type* as the default.

## 23.8.4 MTF Edit =>Settings =>Shape formation toggle

Creating shapes and strings can be a time consuming process so it is possible to set modes for if it occurs or not.



The choices are:

**Don't form shapes or strings** - don't create shapes or strings that are referred to in the **Create=> Shapes** and **Create =>Strings** MTF Modifier commands.

**Form shapes or strings** - generated the shapes and strings as specified in the **Create => Shapes** and **Create =>Strings** MTF Modifier commands.

**Form shapes, not strings** - generated the shapes as specified in the **Create => Shapes** MTF Modifier command but do not generated the strings referred to in the **Create =>Strings** MTF Modifier command.

**Form strings, not shapes** - generated the strings as specified in the **Create => Strings** MTF Modifier command but do not generated the shapes referred to in the **Create =>Shapes** MTF Modifier command.

**OK** - selecting *OK* sets the given *Shape Formation Type*.

## 23.8.5 MTF Edit =>Settings =>Show Extra Start/end?

With MTF Modifier commands, a command applies from the start chainage and finished just before the end chainage. This is to prevent a doubling up in the definitions for the chainage between the end of one command and the beginning of the following command.

To make values change correctly between the MTF definitions at the start and end chainages, for most MTF Modifier commands there are **Extra start** and **Extra end** tick boxes,.

If **Extra start** is ticked on and the command is not the first, a section is created **Extra start/end value before** the Start chainage of the command. Because the definition for this extra section is given by the **previous** command. This ensures that the previous command applies to almost to the this Start chainage.

If **Extra end** is ticked on, a section is created **Extra start/end value** before the End chainage of the command. The definition of this extra section is given by the current command. This ensures that the current command applies almost to the end, especially when it is the last command and there is no next Start chainage.

For most situations, the **Extra start** and **Extra end** tick boxes for an **MTF Modifier Command** can be left ticked on and the default for previous versions (and V11) is to have them both ticked on.

In V11 it is possible to **not show** the **Extra start** and **Extra end** tick boxes in the grid for the **Left and Right MTF Modifiers** panels or on the Command panels, by using the MTF setting **Show Extra start/end?** and **Show extra start/end column?**.

If **Show Extra start/end?** is **ticked**, the **Extra start** and **Extra end** tick boxes are shown on the **MTF Modifier command** panels.

If **Show Extra start/end?** is **not ticked**, the **Extra start** and **Extra end** tick boxes are **not** shown on

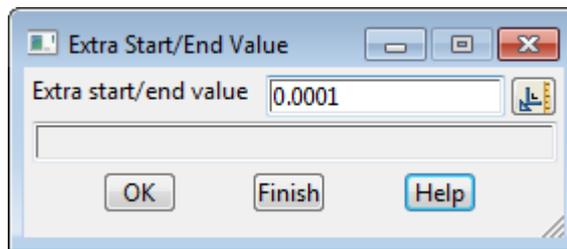
the **MTF Modifier command panels**

**Notes:**

1. The display of the **Extra start** and **Extra end** columns in the **Left/Right Modifiers** panel is controlled by the [23.8.7 MTF Edit =>Settings =>Show extra start/end column?](#).
2. The **Extra start/end value** is set by [23.8.6 MTF Edit =>Settings =>Extra Start/end value](#).

## 23.8.6 MTF Edit =>Settings =>Extra Start/end value

For **12d Model** itself, the **Extra start/end value** does not need to be changed but when writing out sections that are to be loaded into some software products, the sections sometimes can't be closer than a fixed value in the other software. In that case modifying the **Extra start/end value** will allow you to get over that restriction.



**Extra start/end value** - the distance to use for placing extra sections before the start and end chainages.

**OK** - selecting **OK** sets the **Extra start/end value**.

## 23.8.7 MTF Edit =>Settings =>Show extra start/end column?

If **Show extra start/end column?** is **ticked**, the **Extra Start** and **Extra End** columns are shown in the **Left/Right Modifiers** panel.

If **Show extra start/end column?** is **not** ticked the **Extra Start** and **Extra End** columns are **not** shown in the **Left/Right Modifiers** panel.

Note that the **Extra start/Extra end** tick boxes appearing in the **MTF Modifier Commands** is controlled by the [23.8.5 MTF Edit =>Settings =>Show Extra Start/end?](#).

## 23.8.8 MTF Edit =>Settings =>Show absolute column?

If **Show absolute column?** is **ticked**, the **Absolute** column is shown in the **Left/Right Modifiers** panel.

If **Show absolute column?** is **not** ticked the **Absolute** column is **not** shown in the **Left/Right Modifiers** panel.

Note that in both cases, the **New value usage** field (which the **Absolute** column represents) is shown in the **MTF Modifier Link command panels**.

## 23.8.9 MTF Edit =>Settings =>Show interval column?

If **Show interval column?** is **ticked**, the **Interval** column is shown in the **Left/Right Modifiers** panel.

If **Show interval column?** is **not** ticked the **Interval** column is **not** shown in the **Left/Right**

**Modifiers panel.**

Note that in both cases, the **Interval** field is shown on the **MTF Modifier** command panels.

### 23.8.10 MTF Edit =>Settings =>Hinge link name

The name of the link for the Hinge string can now be given a user defined name.

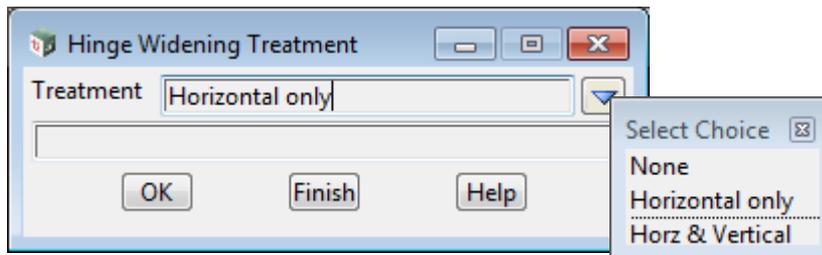


**Hinge name** - the name used for the hinge link.

**OK** - selecting **OK** sets the **Hinge name**.

### 23.8.11 MTF Edit =>Settings =>Hinge widening treatment

The **Hinge Widening Treatment** specified in the **Hinge Widening Treatment** panel.



The choices are:

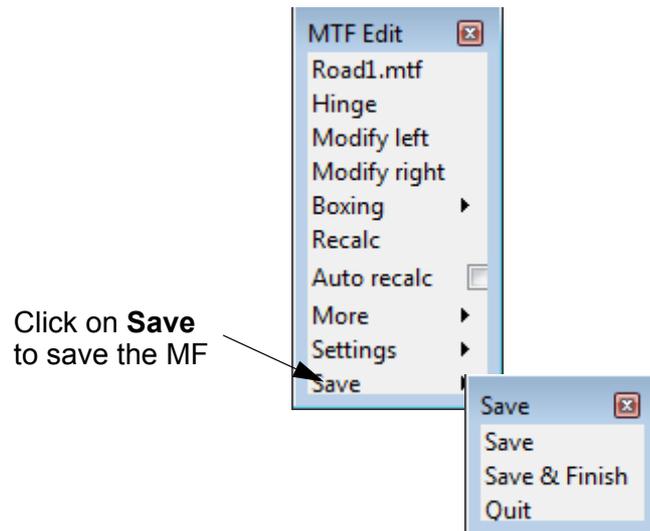
**None** -

**Horizontal only** -.

**Horizontal & Vertical** -.

**OK** - selecting **OK** sets the **Treatment**.

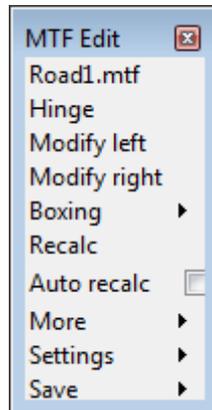
## 23.9 MTF Edit => Save



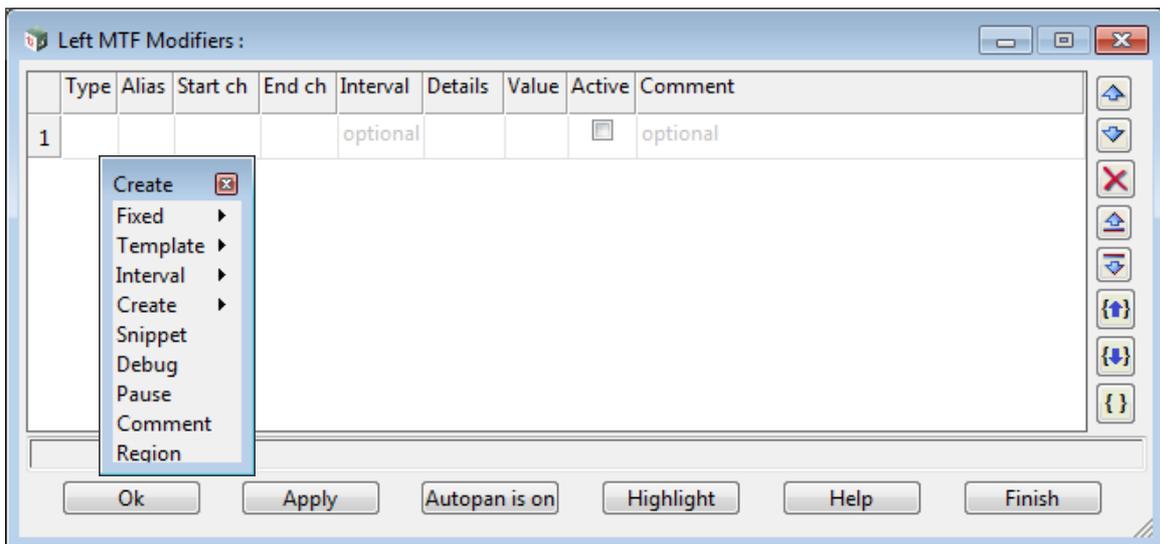
Clicking **Save** on the **MTF Edit** menu (without walking right) saves the MTF file.

Apart from being in a new menu position, **Save**, **Save & Finish** and **Quit** on the **Save** menu work the same as they did in **12d Model 10**.

# 24. MTF Edit => Modify Left/Right



Clicking on **Modify left/right** brings up the **Left/Right Modifiers** panel.



The **Left/Right Modifiers** panels now have **Tick boxes** in the Grid, **Aliases**, **Regions** plus a **Highlights** button.

There are now layers and shapes defined, and the **Create** menu has been totally revamped.

**Absolute** in the *MTF Modifier Commands* has been replaced by a choice box **New value usage** to give the user more information about it actually means but to save space, the column in the grid is still labelled **Absolute**.

See

- [24.1 MTF Links, Points, Sections, Strings and Trimeshes .](#)
- [24.2 Tick Boxes in the Grid .](#)
- [24.3 Turning Off Extra Start/End .](#)
- [24.4 Env Variable to Turn Off Interval Column .](#)
- [24.5 Env Variable to Turn Off Absolute Column .](#)
- [24.7 New Look for MTF Modifier Panels .](#)
- [24.8 Alias for Left and Right Modifiers .](#)
- [24.9 Absolute Replaced by New Value Usage .](#)

[24.10 Regions .](#)

[24.11 Highlight Button .](#)

[24.12 MTF Left/Right Modifiers Create Menu .](#)

[24.13 Defining and Using Snippets](#)

## 24.1 MTF Links, Points, Sections, Strings and Trimeshes

See

[24.1.1 MTF Links, Points and Strings](#)

[24.1.2 MTF Links and Layers](#)

[24.1.3 MTF Shapes and Trimeshes](#)

## 24.1.1 MTF Links, Points and Strings

A **Modifiers and Template File (MTF)**, like a Template, is defined in terms of the links which go together to make up an **MTF section**.

As for a **12d Template**, an **MTF Section** consists of a **left hand side** ending at the **Hinge Point**, followed by a **right hand side**.

Each side is made up of a number of **MTF links** and each link is defined by any two of the three

- (a) width of the link
- (b) change in z-value from the beginning of the link to the end of the link. that is, the z-value at the end of the link minus the z-value at the start of the link. This is referred to as the **height** of the link.
- (c) xfall or slope

and the links are connected sequentially to form a cross section.

Notice that the height is only a **relative** height, **not** a z-value.

On the **left hand side**, the MTF links go from right to left. The vertex (point) at the end of the link is given the same name as the link. The far right of the left hand side is the Hinge Point.

On the **right hand side**, the MTF links go from left to right. The vertex (point) at the end of the link is given the same name as the link. The far left of the right hand side is also the Hinge Point.

So the definition of the MTF Section is not given in absolute offset and absolute height but each **MTF link** is defined **relative** to the **previous link** and the left and right hand sides are positioned around a **Hinge Point**.

Each **MTF link** has a **name** and **colour** and it is best for all the links in an MTF section to have unique names.

The **MTF point** at the **end** of the **MTF link** is given the **name** of the **link** and is called a **MTF Point**.

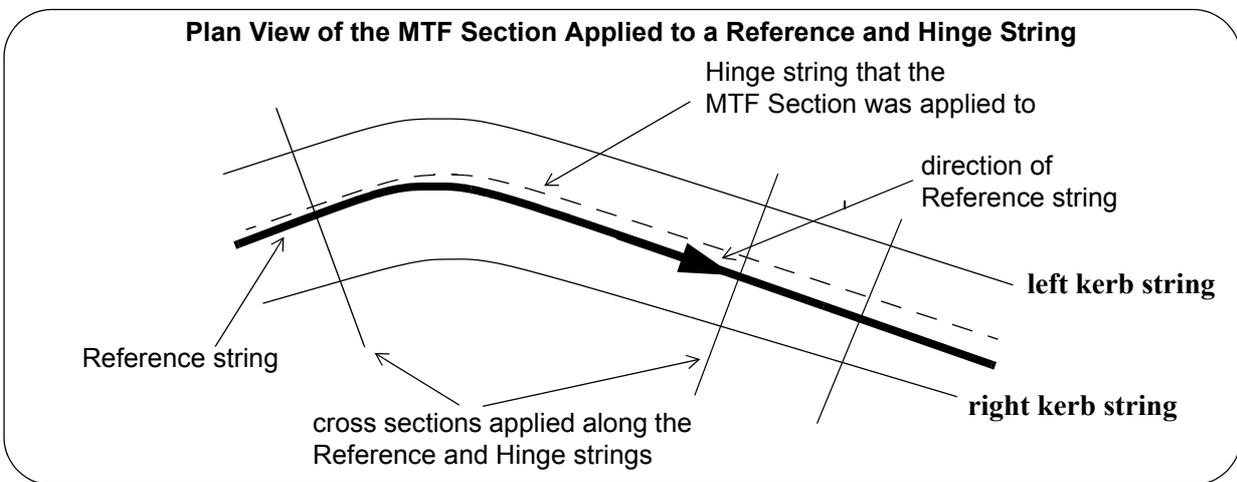
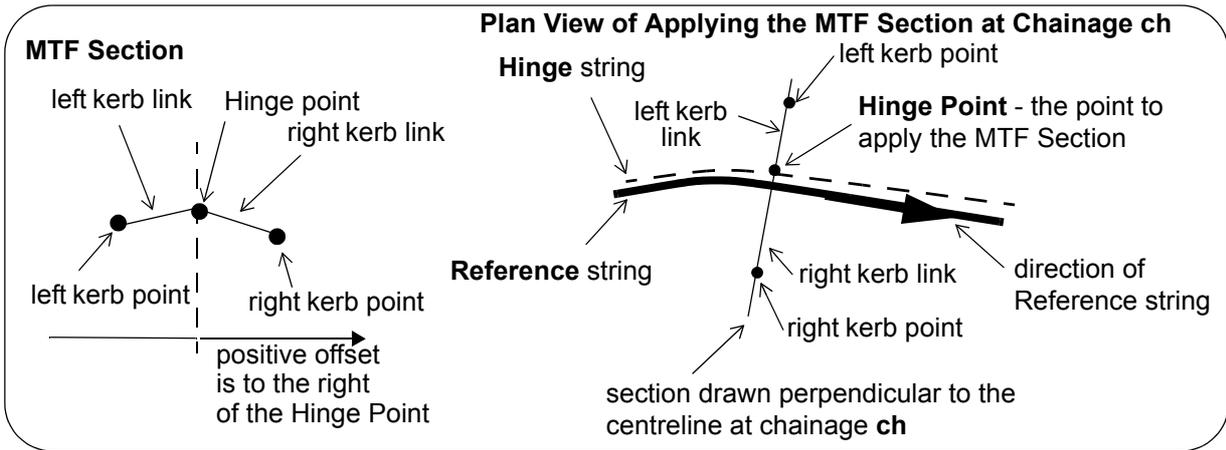
So on the left hand side of the Hinge Point, the MTF Point is at the left hand end of the link and on the right hand side of the Hinge Point, the MTF point is at the right hand end of the link.

When the **MTF** is **Applied** to a **Reference** and a **Hinge String**, **MTF links and links** are created at each chainage, and also whether or not, the **same MTF link and point exists** at the **next chainage**.

So applying the **MTF** defines not only each *MTF link* and but also how each *MTF point* is connected **longitudinally** along the Reference and Hinge strings to form a string, **or more than one string** if there are some chainages where the *MTF point* does not exist.

The longitudinal strings (**MTF strings**) created by a *MTF point* are given the same name and colour as the *MTF point*.

As an example, the MTF with one link called *left kerb* and one link on the right called *right kerb* applied down the a Reference and Hinge String with produce two strings called *left kerb* and *right kerb*:



For brevity, when there is no confusion, the longitudinal MTF strings created by the MTF are often just called **e strings**.

Unlike the simpler *Apply Template*, the **Apply MTF** option does not automatically create strings for all the *MTF points* but allows the user to specify which strings are created. See [24.12.9.2 Left/Right Modifiers => Create =>Create =>Strings](#)

Continue to the next section [24.1.2 MTF Links and Layers](#) or return to [24.1 MTF Links, Points, Sections, Strings and Trimeshes](#)

## 24.1.2 MTF Links and Layers

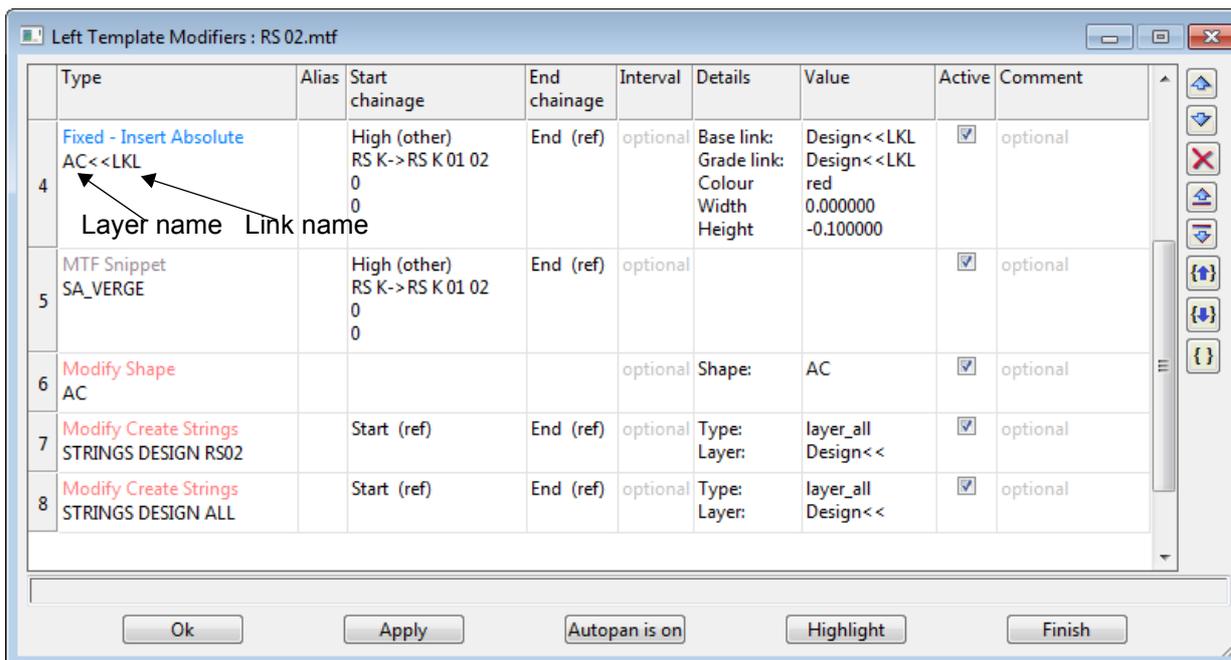
As well as having a name, each **MTF Link** also has a unique **Layer**. The default **Layer** is **Design**.

**Layers** are simply a way of grouping certain links together so that they can be referred to as one thing. For example, the default Layer **Design** is usually where all the links go that make up the Design.

When a MTF Link is created, the Layer is given at the same time as the link name and colour and most MTF commands work for any Layer.

A **MTF Link** of name *link\_name* in the layer *layer\_name* is denoted by:

***layer\_name << link\_name***



In the **Apply MTF Function**, strings or sections are only automatically created for the layer **Design**.

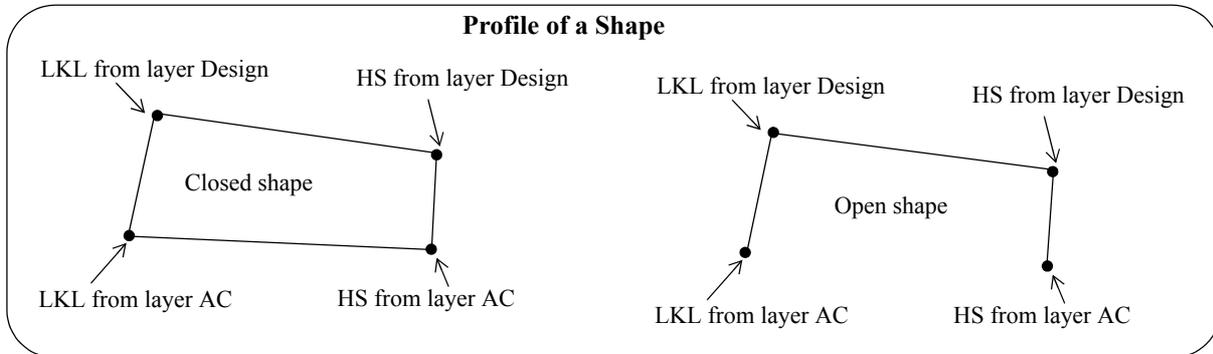
For the other Layers, strings are created using an MTF command. See [24.12.9.2 Left/Right Modifiers => Create =>Create =>Strings](#).

Continue to the next section [24.1.3 MTF Shapes and Trimeshes](#) or return to [24.1 MTF Links, Points, Sections, Strings and Trimeshes](#).

### 24.1.3 MTF Shapes and Trimeshes

A **Shape** is made up of a group of **MTF points** that in profile are joined together in a specific order. Each **MTF point** can come from any **Layer**.

A Shape can be open or closed

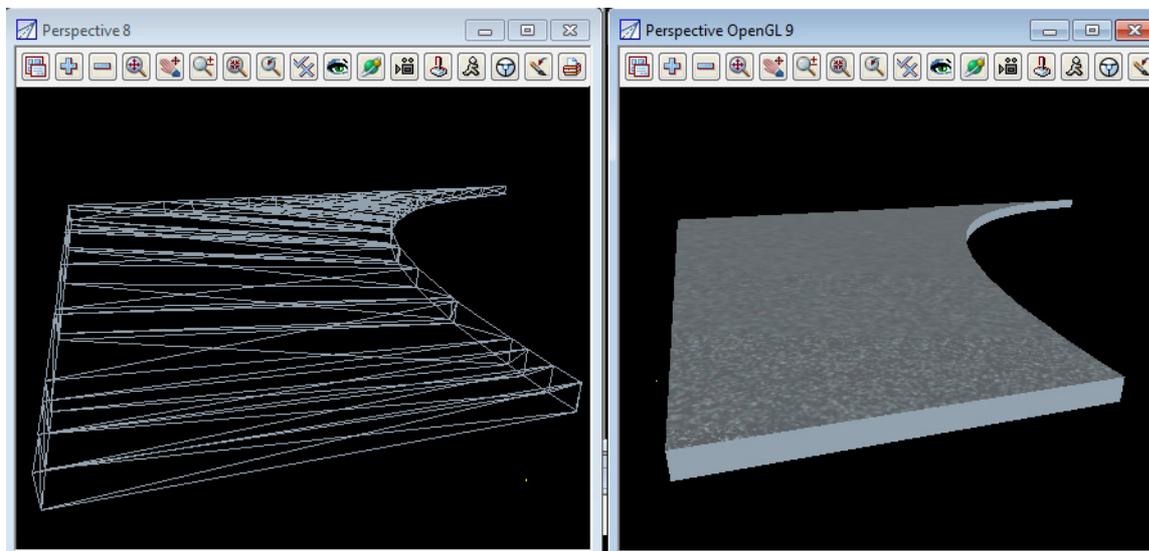


So shapes are similar to cross sections except a shape can go back under or over itself, and can be closed.

When an **MTF** is Applied to a Defence and Hinge string (**Apply MTF Function**), the Shape will be defined at chainages along the Reference string and Hinge string, and the MTF Points of the Shape can generate MTF strings for the Shape.

A **Trimesh** can be constructed from the MTF sections and MTF strings generated by applying the Shape along the Reference and Hinge string.

So the **Trimesh of a Shape** can be thought of as a profile swept (extruded) along the Reference and Hinge string. But remember that at any chainage, the **MTF** can change the positions of each MTF point so the profile of the Shape can change from chainage to chainage.



There is a **MTF** option to create a Trimesh of a Shape and its *trimesh*. See [24.12.9.1 Left/Right Modifiers => Create =>Create =>Shapes](#)

Return to [24.1 MTF Links, Points, Sections, Strings and Trimeshes](#)

## 24.2 Tick Boxes in the Grid

The Grid had been upgraded and there are now tick boxes instead of **Yes/No** choices in the **Absolute**, **Extra start**, **Extra end** and **Active** columns.

## 24.3 Turning Off Extra Start/End

In V11 it is possible to not display the **Extra start** and **Extra end** tick boxes on the **MTF Command** panels and not display the **Extra start** and **Extra end** columns in the **Left/Right Modifiers** panel.

This is controlled by the *MTF Setting* **SHow Extra Start/End?**. See [23.8.5 MTF Edit =>Settings =>Show Extra Start/end?](#).

## 24.4 Env Variable to Turn Off Interval Column

There is an environment variable

`NEW_MTF_EDITOR_SHOW_INTERVAL_COLUMN_4D`

and when it has value zero, the **Interval** column is not displayed in the Grid.

In the **Edit Environment Variables** panel brought up by the option

**Project =>Management =>env.4d**

the tick box of the env variable is on the page

Env.4d >MTF & Boxing >MTF Editor General

and is called **Editor, show interval column**.

Note that for each individual MTF there is now a setting in **MTF Edit** to turn the showing of the Interval column on and off for that MTF. So this environment variable is probably best left set to 1.

## 24.5 Env Variable to Turn Off Absolute Column

There is an environment variable

`NEW_MTF_EDITOR_SHOW_ABSOLUTE_COLUMN_4D`

and when it has value zero, the **Absolute** column is not displayed in the Grid.

In the **Edit Environment Variables** panel brought up by the option

**Project =>Management =>env.4d**

the tick box of the env variable is on the page

Env.4d >MTF & Boxing >MTF Editor General

and is called **Editor, show absolute column**.

## 24.6 Layer and Link Under Command Name

In the **Left** and **Right Modifiers** panel, the major piece of information is now placed under the command name in the Type column. It is also written in black.

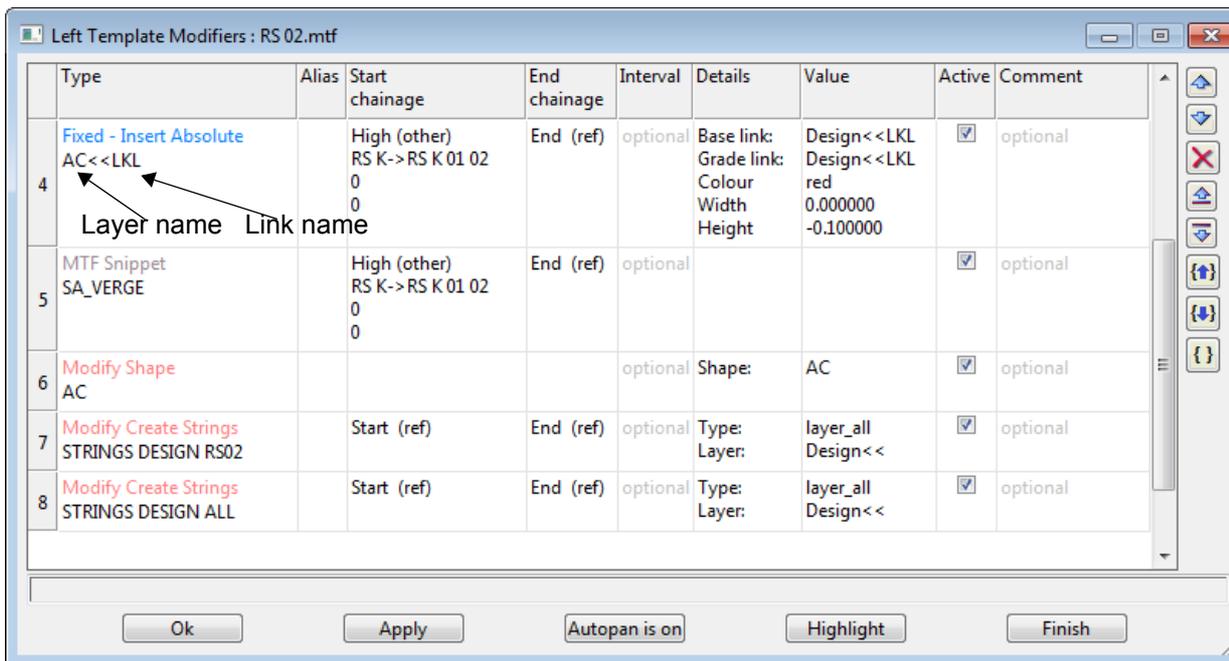
For example,

For **Fixed =>Insert absolute** command, the layer and link name are written under the command name **Fixed - Insert Absolute**.

For **Fixed =>Snippet**, the Snippet name is written under the command name **MTF Snippet**.

For **Fixed =>Shape**, the name of the shape is written under the command name **Modify Shape**.

For **Fixed =>Create string**, the name of the model for the strings is written under the command name **Modify Create Strings**.



## 24.7 New Look for MTF Modifier Panels

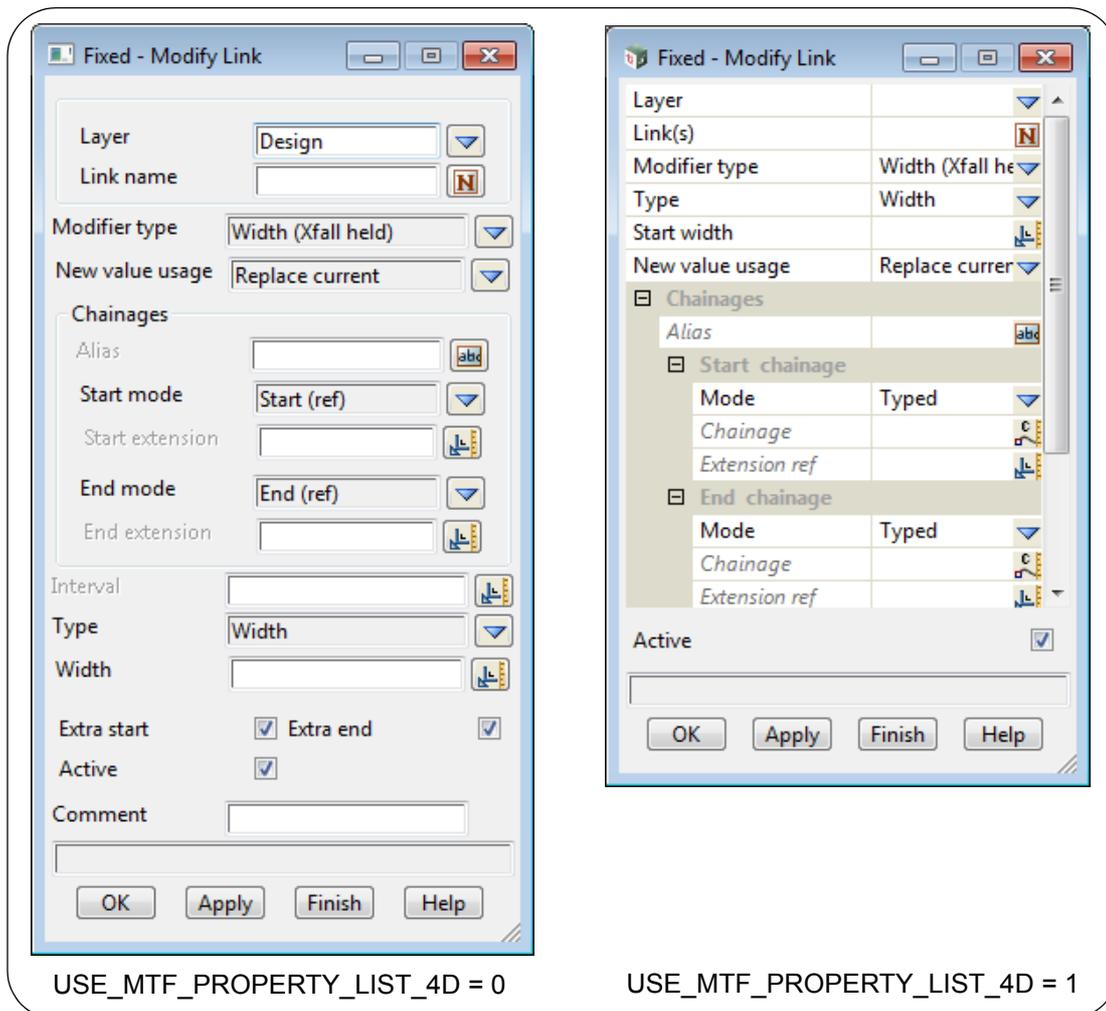
In V11 the panels for the **Left** or **Right Modifiers** have been changed to use what is know as Property Lists and look different to V10.

The new look is on by default but can be controlled by the environment variable

USE\_MTF\_PROPERTY\_LIST\_4D

If USE\_MTF\_PROPERTY\_LIST\_4D = 1, the new property lists are used in the MTF Modifier panels.

If USE\_MTF\_PROPERTY\_LIST\_4D = 0, the new property lists are **not** used in the MTF Modifier panels and the panels will have the same look as in V10.



**Note:**

There is no special field for this environment variable in the **Edit Environment Variables** panel so the set the value to 0 instead of using the default value of 1, you need to add the environment variable USE\_MTF\_PROPERTY\_LIST\_4D and its value 0 to the **Variables** section of the **Edit Environment Variables** panel.

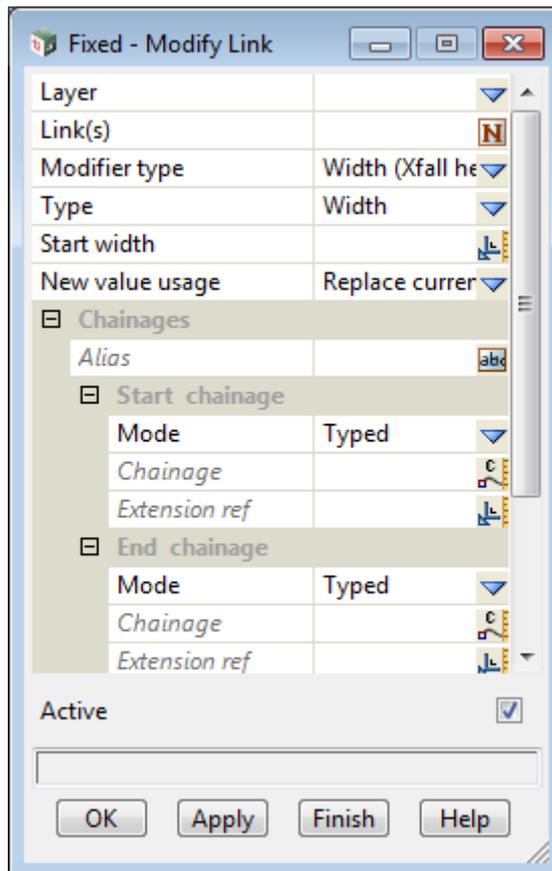
## 24.8 Alias for Left and Right Modifiers

For each *Left/Right Modifiers* Command, there is now an optional text field called **Alias**.

If an **Alias** is given for a command in the **Left Modifiers** panel then it must be unique amongst all the Aliases for the **Left Modifiers** commands.

Similarly if an **Alias** is given for a command in the **Right Modifiers** panel then it must be unique amongst all the Aliases for the **Right Modifiers** commands.

Hence an **Alias** can be used to uniquely refer to a particular command in the **Left** or **Right Modifiers** grid **independently** of its row number.



**Alias** text box

*if not blank, a text name that can be used to refer to the row in the grid for this command. The name must unique amongst all the Aliases for the **Left Modifier** commands if this command is in the **Left Modifiers** panel, or unique amongst all the Aliases for the **Right Modifier** commands if this command is in the **Right Modifiers** panel.*

*The **Alias** is a reference to this command in the **Left/Right Modifiers** grid that can be referred to by Smart Chainages used in the **Left/Right Modifiers** commands in the grid*

An **Alias** can be used in the Smart Chainage modes, Relative to Alias Start/End. See [9.4.5 Relative to Alias Start](#) and [9.4.6 Relative to Alias End](#).

## 24.9 Absolute Replaced by New Value Usage

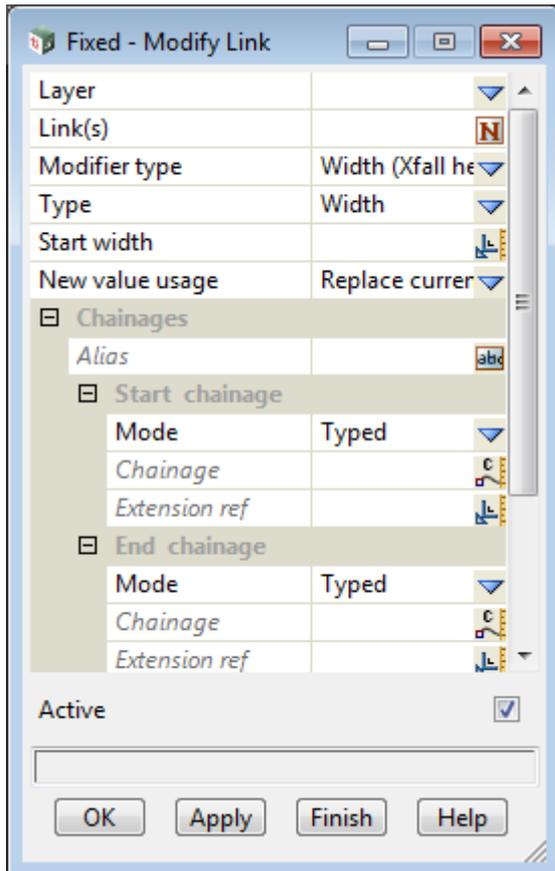
**Absolute** in the Modifier commands

**Fixed** => **Link**

**Cut** => **Link**

**Fill** => **Link**

has been replaced by a choice box **New value usage**. And **New value usage** has been placed immediately after **Modifier type**.



**New value usage**

choice box

Add to the current value of the link,  
Replace the current value of the link

*if **Add to the current value of the link**, the value of the parameter at each chainage is **added to the current value for the parameter**. This is the replacement for **Absolute not ticked**.*

*if **Replace the current value of the link**, the value of the parameter at each chainage **replaces the current value for the parameter**. This is the replacement for **Absolute ticked**.*

## 24.10 Regions

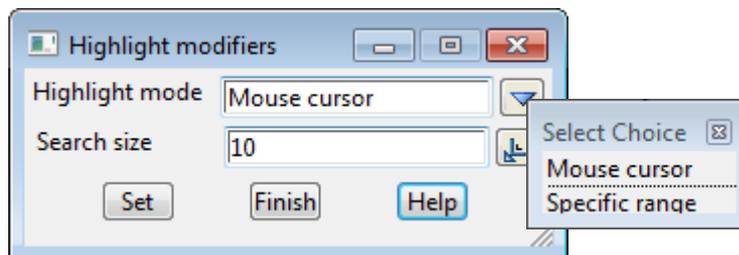
Regions are now available as a command in the **Left/Right Modifiers** grid.

There are **Regions**, **Previous Region** and **Next Region** icons at the right hand end of the **Left/Right Modifiers** panel.

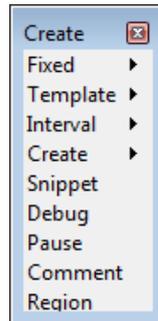
See [12.11 Regions in Some Grids](#).

## 24.11 Highlight Button

There is now a **Highlight** button and when it is clicked brings up the **Highlight Modifiers** panel.



## 24.12 MTF Left/Right Modifiers Create Menu



To make it clearer which options allow multiple link names, such a field in a MTF panel is now called **Link(s)**. See [24.12.3 Link or Link\(s\)](#).

A major change for many of the options in the **Fixed**, **Cut** and **Fill** sections of the **Create** menu is that how the link was originally defined in terms of width, height, xfall or slope is no longer important to how the link can be modified. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).

For the changed options in the **Create** menu, see

[24.12.7 Left/Right Modifiers => Create =>Interval](#)

[24.12.4 Left/Right Modifiers => Create =>Fixed](#)

[24.12.5 Left/Right Modifiers => Create =>Template =>Cut](#)

[24.12.8 Left/Right Modifiers => Create =>Snippet](#)

[24.12.10 Left/Right Modifiers => Create =>Debug](#)

[24.12.11 Left/Right Modifiers => Create =>Pause](#)

[24.12.12 Left/Right Modifiers => Create =>Region](#)

### 24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions

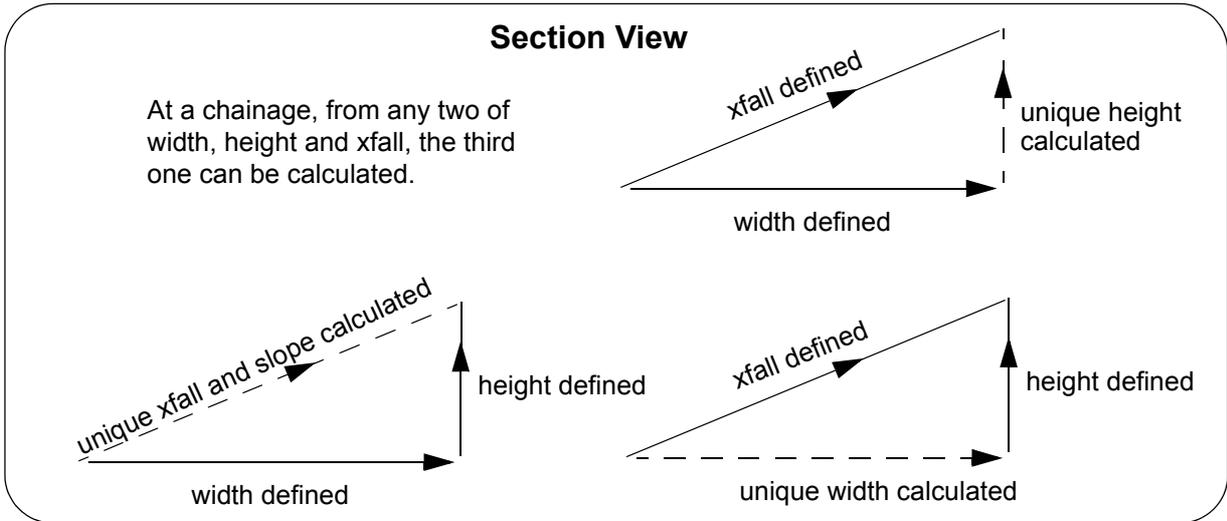
For most of the **Modifier** commands, the command **modifies** the current value of the link of one of width, height, xfall or slope, and uses (**holds**) the current value of the link of another one of width, height, xfall or slope.

Together, the **modified** value and the **held** value defines the new values for the link.

It doesn't matter which two of width, height and xfall or slope were originally used to define the link, the option will internally convert the definition to what is required to be **modified** and **held**.

For example the link may be originally defined by **xfall** and **height** but the link **Modifier type** could be **Modify Width, Hold Xfall** or **Modify Xfall, Hold Width**.

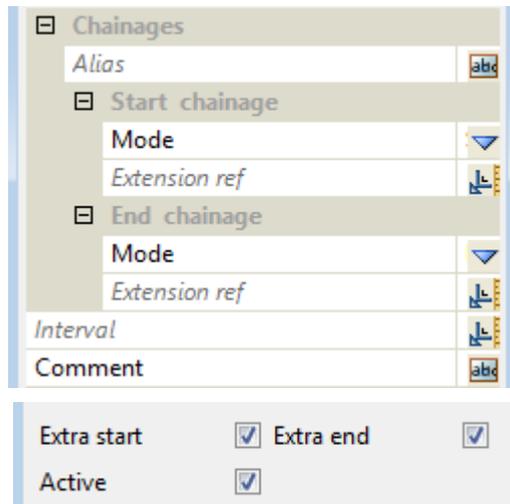
Even though the existing link is not defined by width, for each required chainage, the xfall and height values of the existing link **uniquely define a width** at that chainage. And it is this calculated width that is then modified by **Modify Width, Hold Xfall** or used (held) by a **Modify Xfall, Hold Width**.



So no matter how the link is originally defined, any one of width, height or xfall/slope can be calculated and modified or held by the option.

## 24.12.2 Common Fields on Modifier Panels

Most of the **Modifier** panels have the following common panels fields.



**Alias** text box

*if not blank, a text name that can be used to refer to the row in the grid for this command.*

See [Alias text box](#)

**Start/End chainage** choice box Start (ref)/End (ref)

*defines the start/end chainages for use in the Modifier.*

**Interval** real box

*if not blank, the interval to use to create cross sections and strings over the given chainage range.*

*If blank, the interval given by any **MTF Edit** => **Interval commands**, and/or the **Section separation** value from the **Apply MTF** panel, is used.*

**Extra start** tick box

*if ticked, add an extra x-section 0.1 mm before the start chainage.*

**Extra end** tick box

*if ticked, add an extra x-section 0.1 mm before the end chainage.*

**Comment** text box

*comment to add to the Comment cell in this row of the grid.*

**Active** tick box tick

*if ticked, use this modifier.*

*If not ticked, don't use this modifier.*

**OK** button

*OK stores the values in the fields and removes the panel BUT no recalc is done.*

**Apply** button

*Apply stores the values and leaves the panel on the screen.*

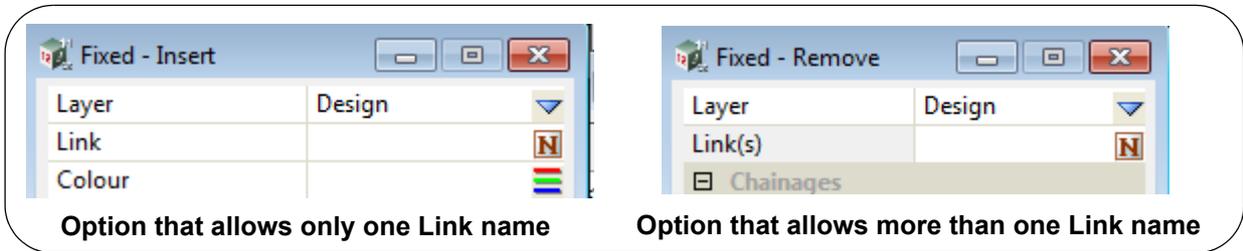
*If the MTF is being used in an **Apply MTF** and **Auto recalc** is ticked in the MTF, then whenever the **Apply** button is clicked, a **recalc** of the associated **Apply MTF** for the MTF is done.*

### 24.12.3 Link or Link(s)

Some **Modifier** panels allow more than one link name to be specified.

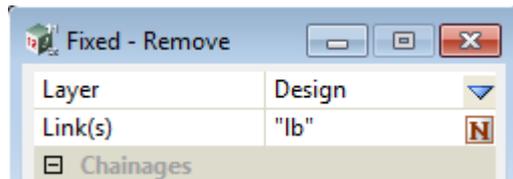
If only one Link is allowed, then the panel field will be **Link**.

If more than one Link is allowed, then the panel field will be **Link(s)**.

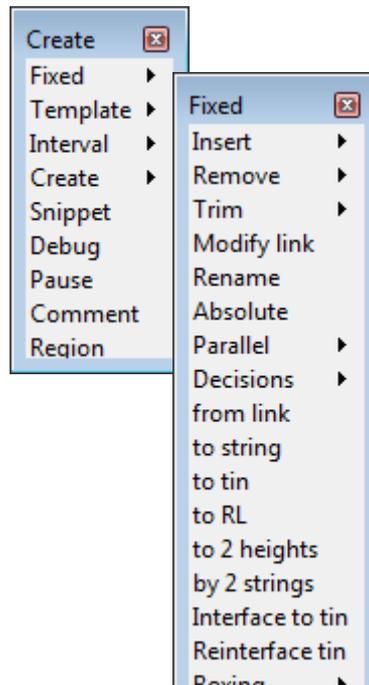


When multiple Links are allowed, each Link name is separated by one or more spaces. If the Link name contains a space then it must be surrounded by double quotes. For example "name with a space".

When an MTF panel that allows multiple Link names is edited, all the Link names will be displayed surrounded by double quotes. This applies even if there is only one Link name in the field.



## 24.12.4 Left/Right Modifiers => Create =>Fixed



For modified options, see

[24.12.4.1 Left/Right Modifiers => Create =>Fixed =>Insert](#)

[24.12.4.2 Left/Right Modifiers => Create =>Fixed =>Remove](#)

[24.12.4.3 Left/Right Modifiers => Create =>Fixed =>Trim](#)

[24.12.4.4 Left/Right Modifiers => Create =>Fixed =>Link](#)

[24.12.4.5 Left/Right Modifiers => Create =>Fixed =>Decisions](#)

[24.12.4.6 Left/Right Modifiers => Create =>Fixed =>from link](#)

[24.12.4.7 Left/Right Modifiers => Create =>Fixed =>to string](#)

[24.12.4.8 Left/Right Modifiers => Create =>Fixed =>to tin](#)

[24.12.4.9 Left/Right Modifiers => Create =>Fixed =>to RL](#)

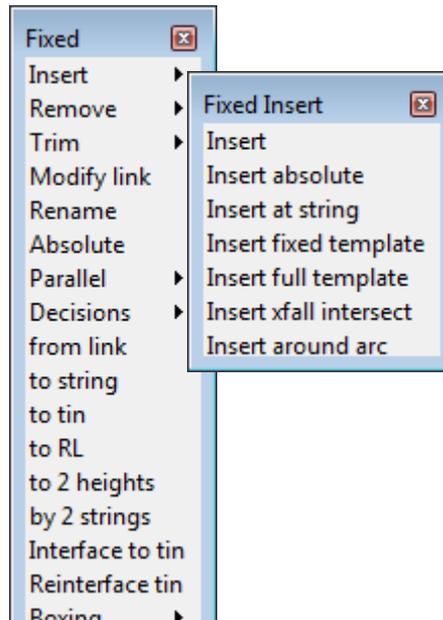
[24.12.4.10 Left/Right Modifiers => Create =>Fixed =>to 2 Heights](#)

[24.12.4.11 Left/Right Modifiers => Create =>Fixed =>by 2 strings](#)

[24.12.4.12 Left/Right Modifiers => Create =>Fixed =>Interface to tin](#)

[24.12.4.13 Left/Right Modifiers => Create =>Fixed =>Boxing](#)

## 24.12.4.1 Left/Right Modifiers => Create =>Fixed =>Insert



See

[24.12.4.1.1 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert](#)

[24.12.4.1.2 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert layer hinge](#)

[24.12.4.1.3 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert at string](#)

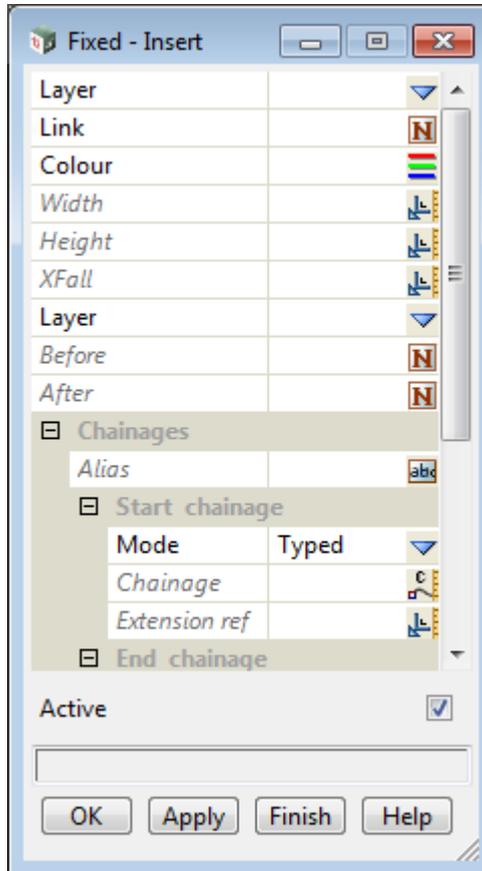
[24.12.4.1.4 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert fixed template](#)

[24.12.4.1.5 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert full template](#)

### 24.12.4.1.1 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert

This is similar to V10 but there is a **After** field.

Also there is a **Layer**, **Alias** and improved smart **Start** and **End** chainage modes.



**Note:** One of **Before** or **After** must be *blank*. If they are both blank then **Before** takes precedence over **After**.

**Layer** choice box available Layers  
*Layer for the new link to go into.*

**Link** text box  
*name for the new link*

**Before** choice box select name menu  
*if **not blank**, the name of the string to insert the new link before.*  
*If **blank** and **After** is **blank**, the link is appended to the end of the fixed part of the template.*

**After** choice box select name menu  
*if **Before** is **not blank**, **After** is ignored.*  
*If **not blank** and **Before** is **blank**, the name of the string to insert the new link after.*  
*If **blank** and **Before** is **blank**, **Before** is used.*

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

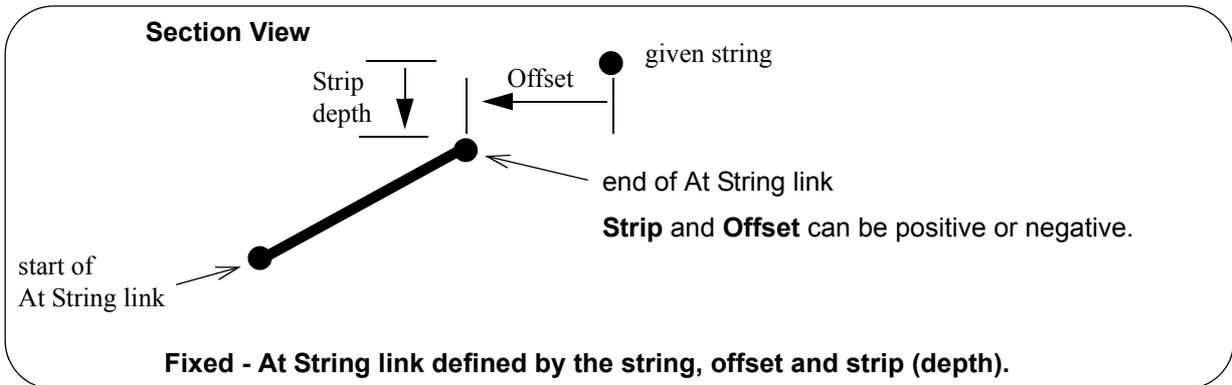
### 24.12.4.1.2 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert layer hinge

Create =>Insert =>Insert layer hinge has not yet been documented.

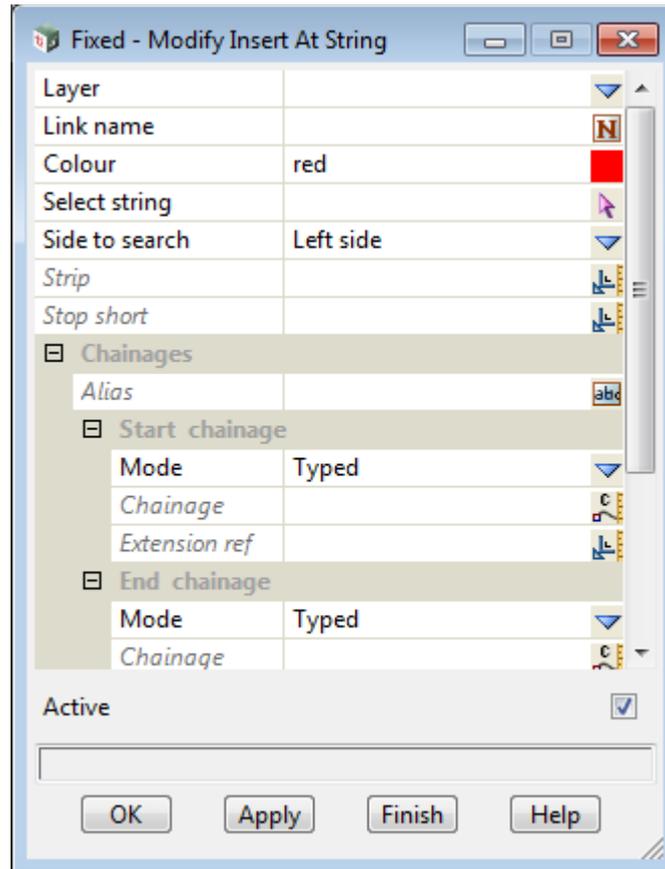
### 24.12.4.1.3 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert at string

The **Insert at string** copies an existing string between the two given chainages and inserts it as a link of the template. The created link can also be given an offset and depth from the string.

Hence the end of the new link is constructed so that it is a given **Offset** from the **string** and stops at the depth **Strip below** the **string**.



This replaces the process in V10 of having to first insert a link for the required chainages and then use the "to String" modifier to move the link onto the selected string. Plus it also allows the **strip** and **offset**.



The new fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>New link type</b> <i>states how the link is to be defined.</i>	choice box	Width and Xfall, Width and Height, Height and Xfall	
<b>Layer</b> <i>Layer for the new link to go into.</i>	choice box		available Layers
<b>Link name</b> <i>name for the new link</i>	text box		
<b>Colour</b> <i>colour for the new link</i>	colour box		available colours
<b>Strip</b> <i>the new link is constructed so that its end is <b>Strip</b> below the selected string. <b>strip</b> is a depth and is positive in the downward direction It can be positive or negative.</i>	real box		measures menu
<b>Offset</b> <i>the new link is constructed so that its end is <b>Offset</b> offset from the selected string. <b>offset</b> is positive when coming back towards the reference string. It can be positive or negative.</i>	real box		measures menu
<b>Select string</b> <i>select the string to copy as a link</i>	string select box		
<b>Side to search</b> <i>side to search to find the string.</i>	choice box	Left side, Right side, Both sides	

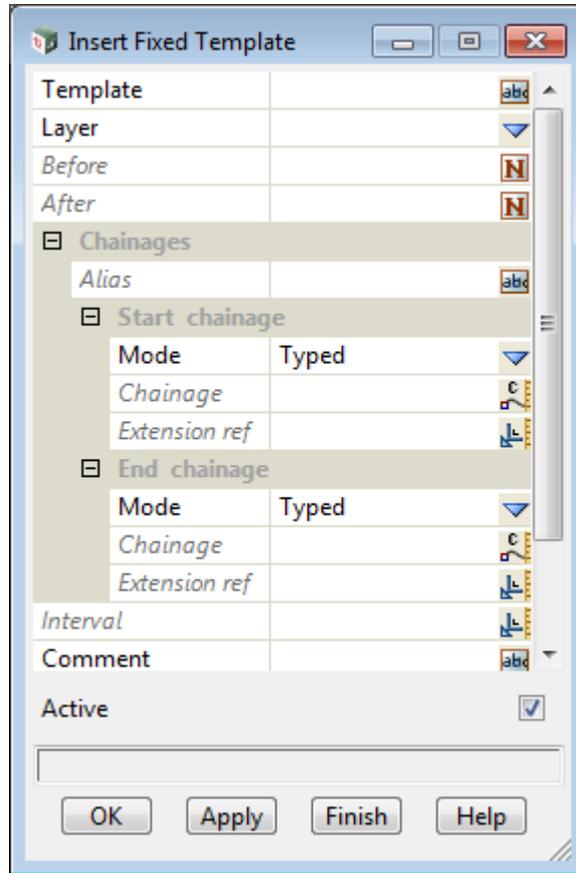
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

24.12.4.1.4 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert fixed template

This is similar to V10 but there is a **After** field.

Also there is a **Layer**, **Alias** and improved smart **Start** and **End** chainage modes.



Only one of **Before** or **After** can be *not blank*. If they are both blank then **Before** takes precedence over **After**.

**Layer** choice box available Layers

*Layer for the links in the fixed part of the template to be inserted into.*

**Before** choice box select name menu

*if not blank, the name of the link to insert the fixed links from the selected template before.  
If blank and After is blank, the fixed links from the selected template are appended to the end of the existing fixed links.*

**After** choice box select name menu

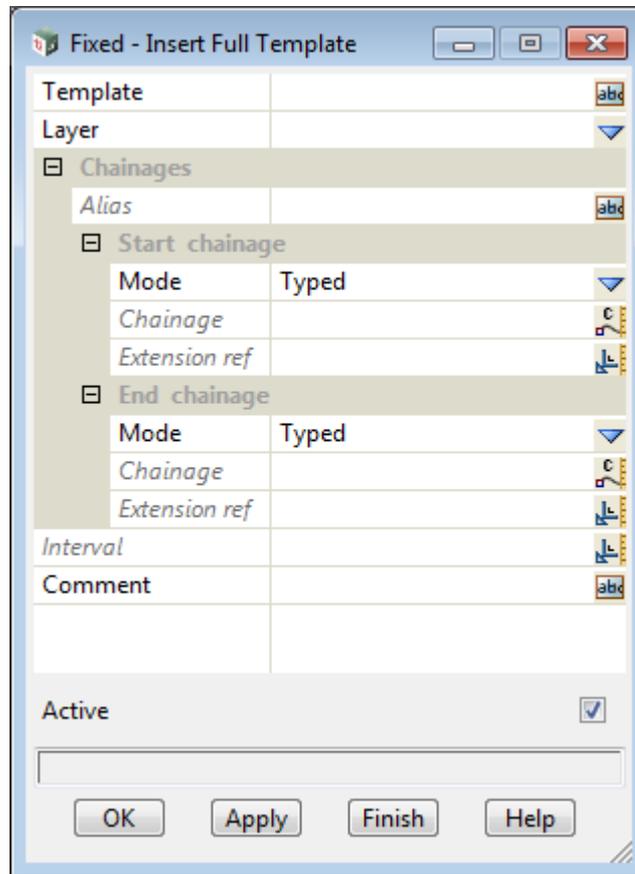
*if Before is not blank, After is ignored.  
If not blank and Before is blank, the name of the link to insert the fixed links from the selected template after.  
If blank and Before is blank, Before is used.*

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

24.12.4.1.5 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert full template

This is similar to V10 but there is a **Layer**, **Alias** and improved smart **Start** and **End** chainage modes.



**Layer** choice box available Layers

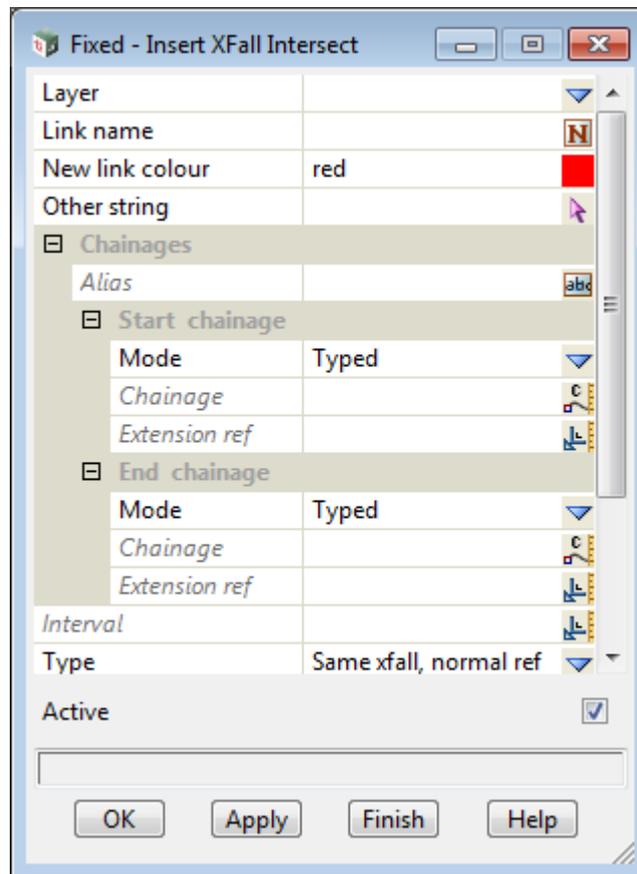
*Layer for the strings in the selected template to be inserted into.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

#### 24.12.4.1.6 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert xfall intersect

This is similar to V10 but there is a **Layer**, **Alias** and improved smart **Start** and **End** chainage modes.



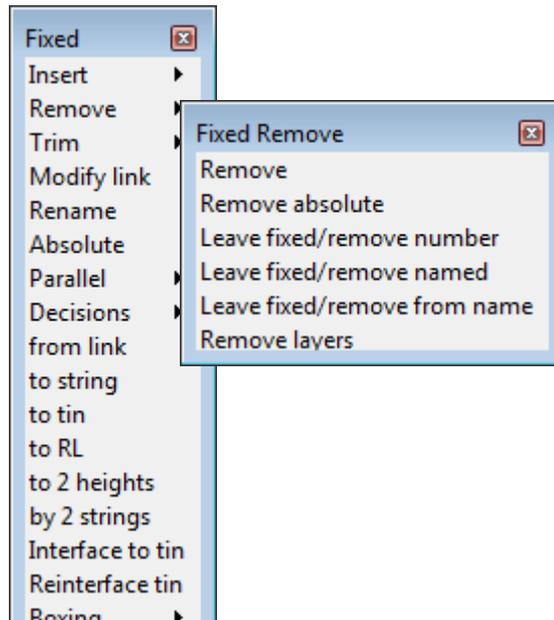
**Layer** choice box available Layers

*Layer for the strings in the selected template to be inserted into.*

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 24.12.4.2 Left/Right Modifiers => Create =>Fixed =>Remove



The first three options are the same as in V10 except that they have a **Layer** field that is defaulted to **Design**.

The options **Leave fixed/remove from name** and **Remove layers** are new.

See

[24.12.4.2.1 Left/Right Modifiers => Create =>Fixed =>Remove => Remove Absolute](#)

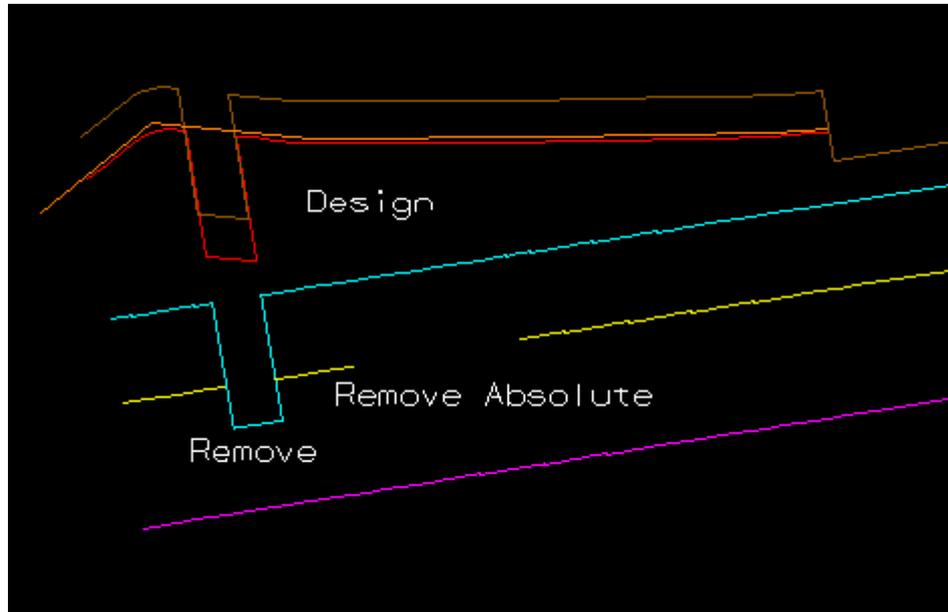
[24.12.4.2.2 Left/Right Modifiers => Create =>Fixed =>Remove =>Leave fixed/remove from name](#)

[24.12.4.2.3 Left/Right Modifiers => Create =>Fixed =>Remove =>Remove Layers](#)

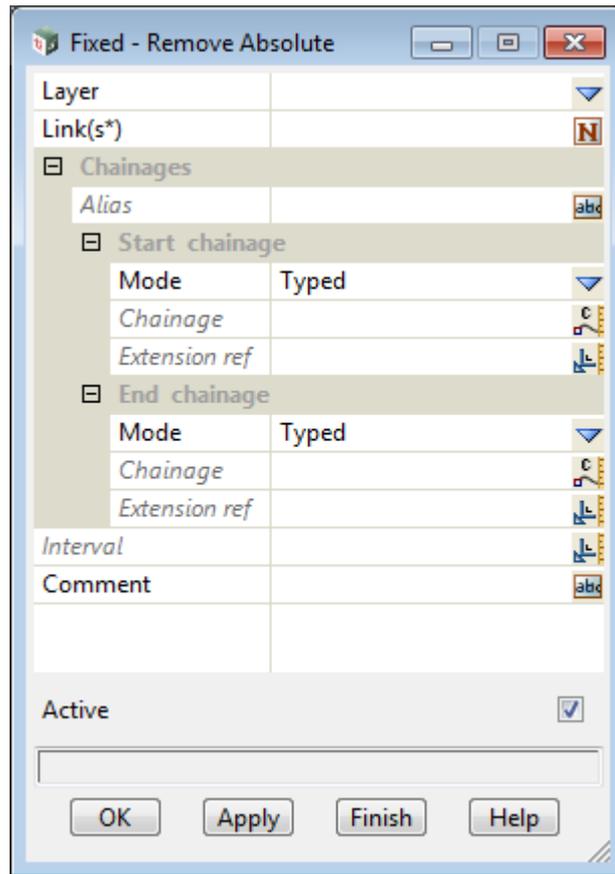
#### 24.12.4.2.1 Left/Right Modifiers => Create =>Fixed =>Remove => Remove Absolute

This option is new in V11.

The **Remove Absolute** option deletes fixed links between given chainages whilst leaving the absolute position of the other links undisturbed.



Selecting **Remove absolute** brings up the **Fixed - Remove Absolute** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b>	layer box		available Layers
<i>Layer to remove the link. For information on Layers.</i>			
<b>Link name</b>	name box		select name menu
<i>name of the link to removed.</i>			

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

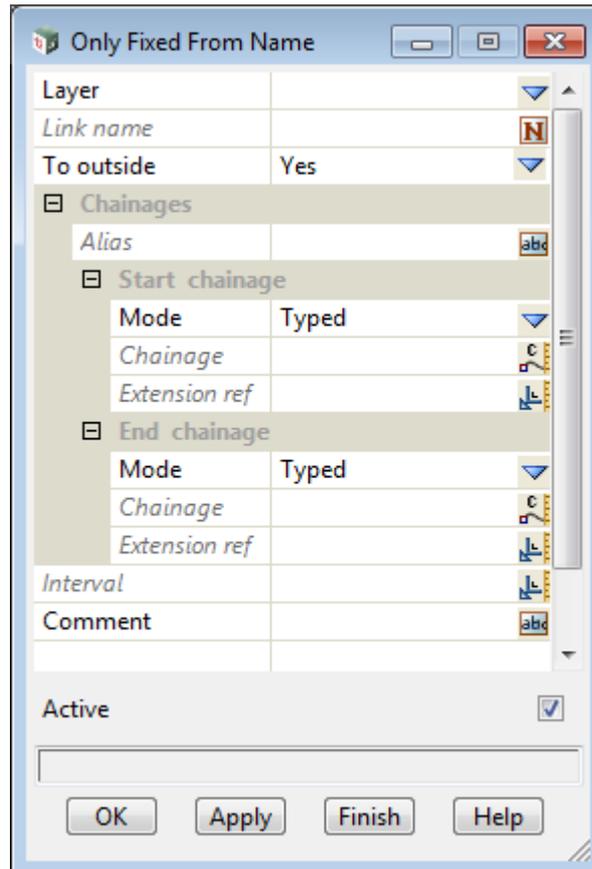
### 24.12.4.2.2 Left/Right Modifiers => Create =>Fixed =>Remove =>Leave fixed/remove from name

This option is new in V11.

The **Leave fixed/remove from name** option removes any of the variable cut and fill, final cut/fill links and decisions that have been defined up to this point in the modifiers grid.

It also removes all the links from a named link, or from a named link to the beginning of the fixed links.

Selecting **Leave fixed/remove from name** brings up the **Only Fixed from Name** panel.



The fields and buttons used in this panel have the following functions.

Field	Description	Type	Defaults	Pop-Up
<b>Layer</b>		choice box		available Layers
<i>Layer to remove links from.</i>				
<b>Link name</b>		text box		
<i>name of the link in the layer to start removing links from/to.</i>				
<b>To outside</b>		choice box		yes, no
<i>if yes, all the links from <b>Link name</b> out are removed. <b>Link name</b> is not removed.</i>				
<i>If no, all the links from the beginning of the Fixed zone until <b>Link name</b> are removed. <b>Link name</b> is not removed.</i>				

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

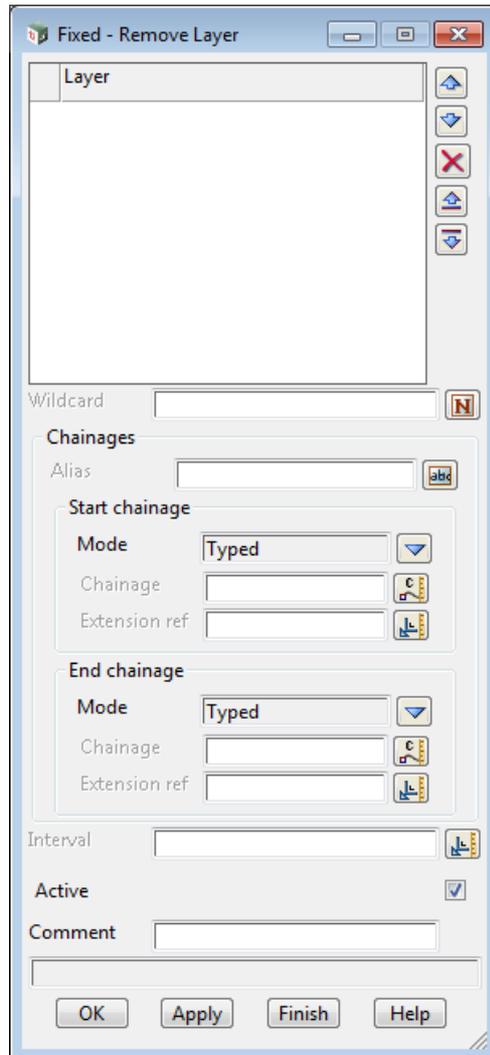
*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.2.3 Left/Right Modifiers => Create =>Fixed =>Remove =>Remove Layers

This option is new in V11.

The **Remove layers** option removes any links in the Layers that have been defined up to this point in the modifiers grid.

Selecting **Remove layers** brings up the **Only Fixed from Name** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

#### Layer Grid

<b>Layer</b>	choice box		available Layers
--------------	------------	--	------------------

*link names that match **Wildcard** are removed from the layers listed in the grid.*

<b>Wild card</b>	text box		
------------------	----------	--	--

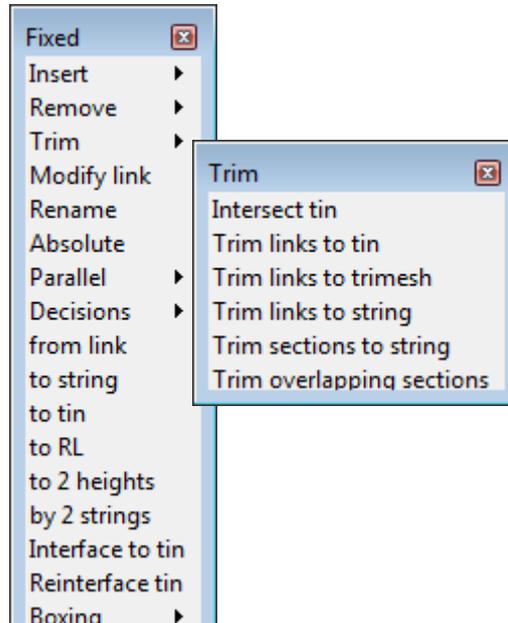
*text including wild cards (\*) and wild characters () that the link names are matched against.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .*

### 24.12.4.3 Left/Right Modifiers => Create =>Fixed =>Trim

These options are new in V11.



See

[24.12.4.3.1 Intersect Tin](#)

[24.12.4.3.4 Trim Links to String](#)

[24.12.4.3.5 Trim Overlapping Sections](#)

### 24.12.4.3.1 Intersect Tin

This option finds the intersection points of the link **Link** within the given vertical tolerance **Intersect z tol** of the stripped **Tin** and creates extra section at nominated chainage distances from the intersection chainages.

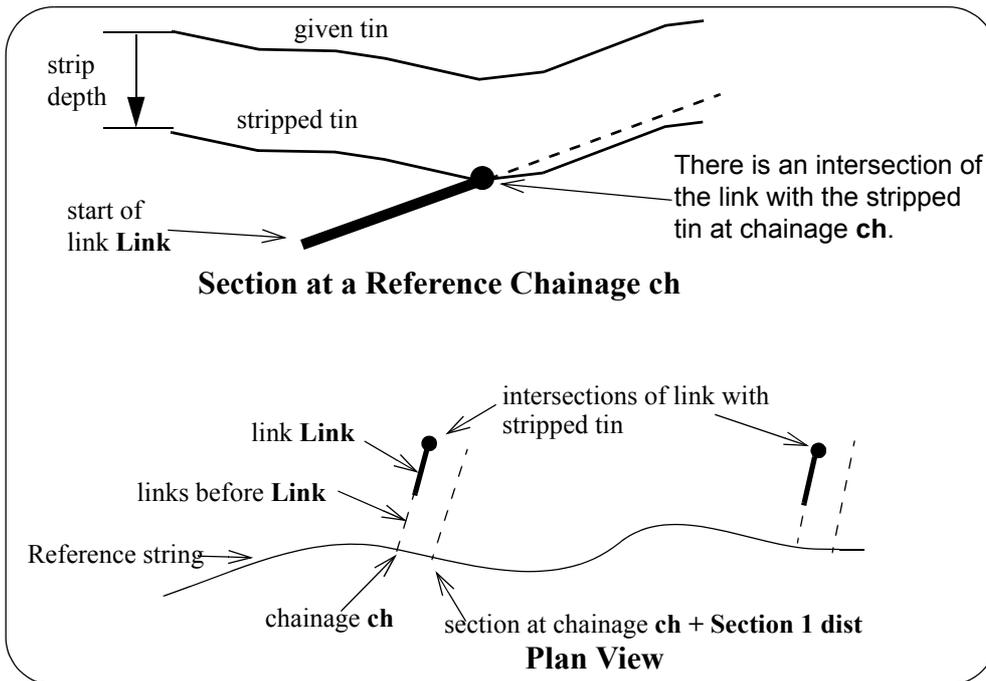
Extra sections are then created at the reference chainages

Reference chainage of intersection of link with (tin - Strip) + **Section 1 dist**

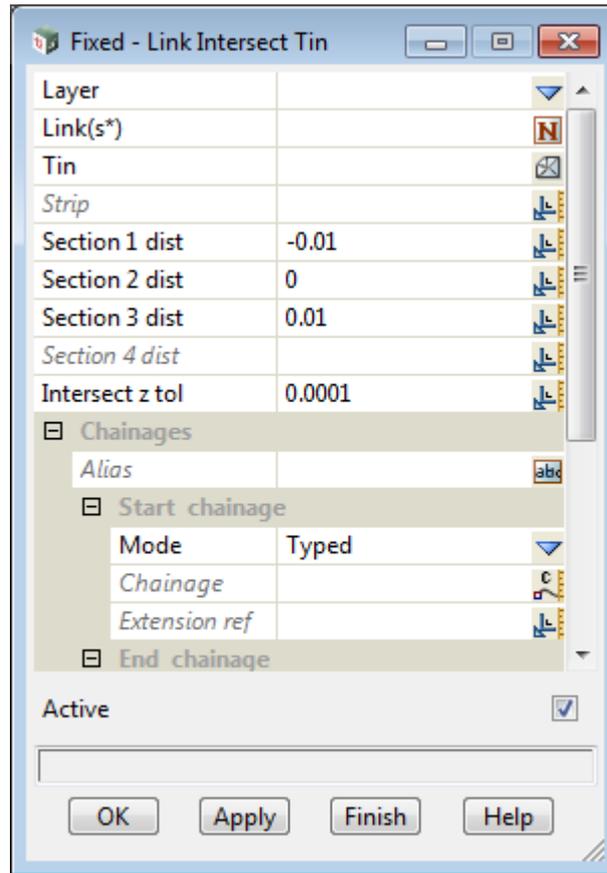
Reference chainage of intersection of link with (tin - Strip) + **Section 2 dist**

Reference chainage of intersection of link with (tin - Strip) + **Section 3 dist**

Reference chainage of intersection of link with (tin - Strip) + **Section 4 dist**



Selecting **Intersect tin** brings up the **Fixed - Link Intersect Tin** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>Layer of the link to cut with the stripped tin.</i>	layer box		available Layers
<b>Link</b> <i>name of the link to cut with the stripped tin.</i>	name box		select name menu
<b>Tin</b> <i>name of the tin to use as the stepped tin to find the intersections with the link <b>Link</b>.</i>	tin box		available tins
<b>Strip</b> <i>the depth to vertically drop the tin before calculating intersections. This is then referred to as the stripped tin.</i>	real box	0	
<b>Section 1, 2, 3, 4 dist</b> <i>if <b>not blank</b>, this value is added to the reference chainages of the intersection of the link with the stripped tin, and new sections are created at this new chainage. The value can be positive, negative and zero.</i> <i>If <b>blank</b>, no section is created for this <b>Section i dist</b>.</i>	real box		
<b>Notes</b>			
1. There may be more than one reference string chainage where the link intersects the stripped tin.			
2. A section is only produced at the intersections of the link with the stripped tin if one of <b>Section i dist</b> is zero.			

**Intersect z tol**                      real box                      0.0001

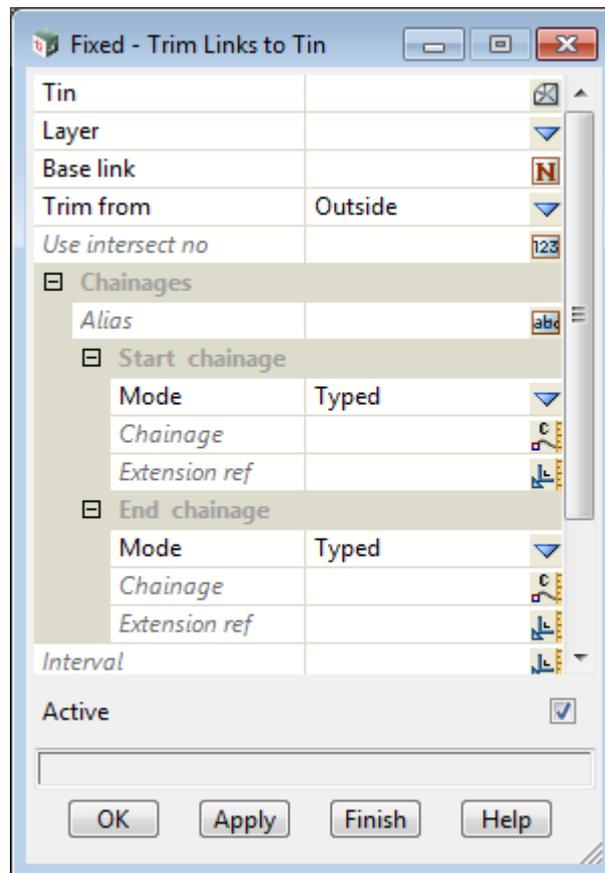
*the intersection of the link with the stripped tin can occur within this vertical distance of the striped tin.*

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .*

### 24.12.4.3.2 Trim Links to Tin

Selecting Trim links to tin brings up the Fixed - Trim Links to Tin panel



The fields and buttons used in this panel have the following functions.

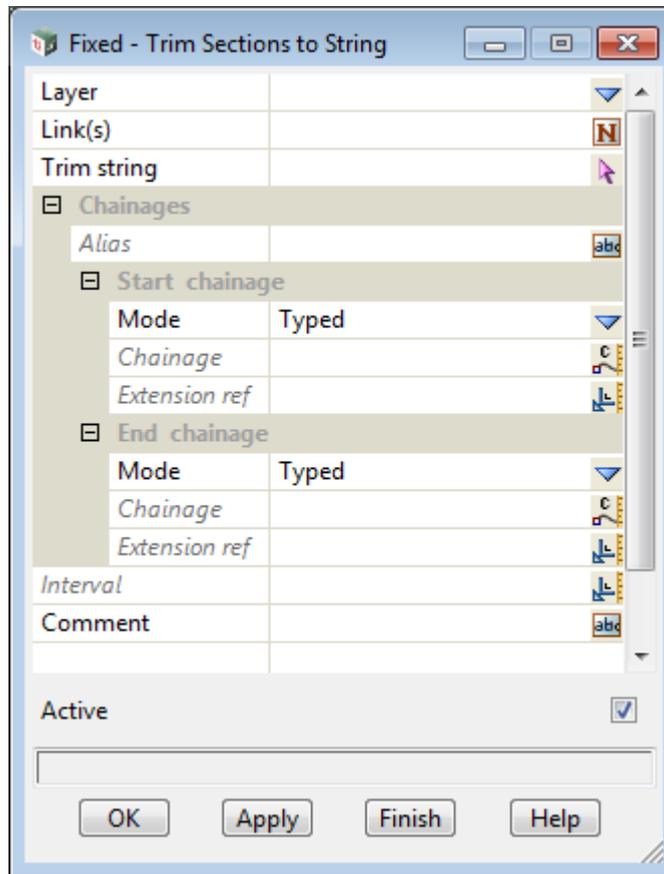
Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>Layer of strings to trim.</i>	layer box		available Layers
<b>Link</b> <i>name of the link to trim against the tin.</i>	name box		select name menu

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.3.3 Trim Sections to String

Selecting Trim sections to string brings up the **Fixed - Trim Sections to String** panel



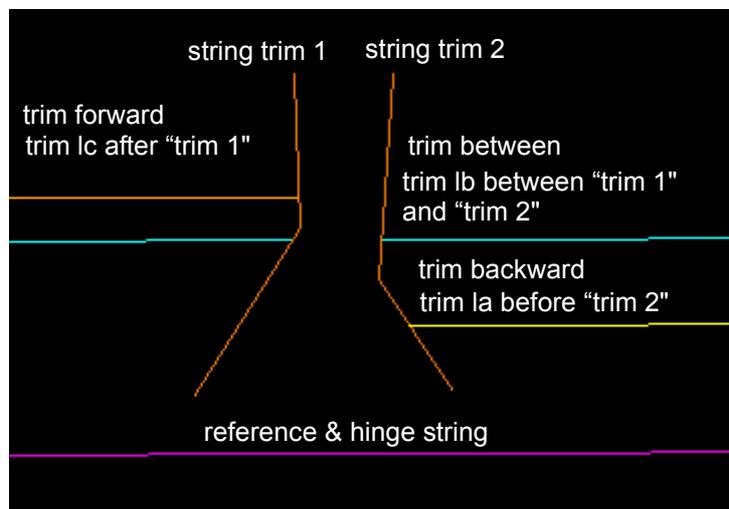
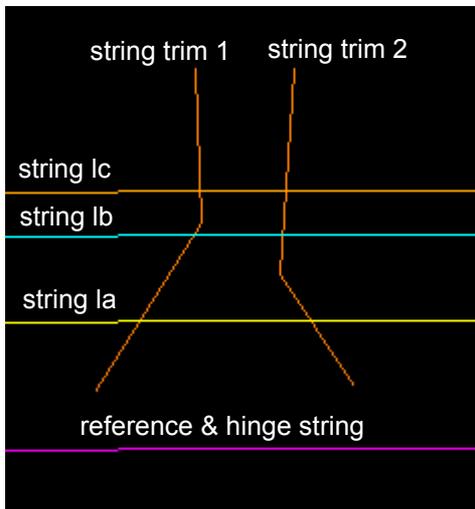
The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>Layer of strings to trim.</i>	layer box		available Layers
<b>Link(s)</b> <i>name of the link to trim against the tin.</i>	name box		select name menu

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**  
 For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

### 24.12.4.3.4 Trim Links to String

This option trims links before a selected string, or after a selected string, or between two selected strings.

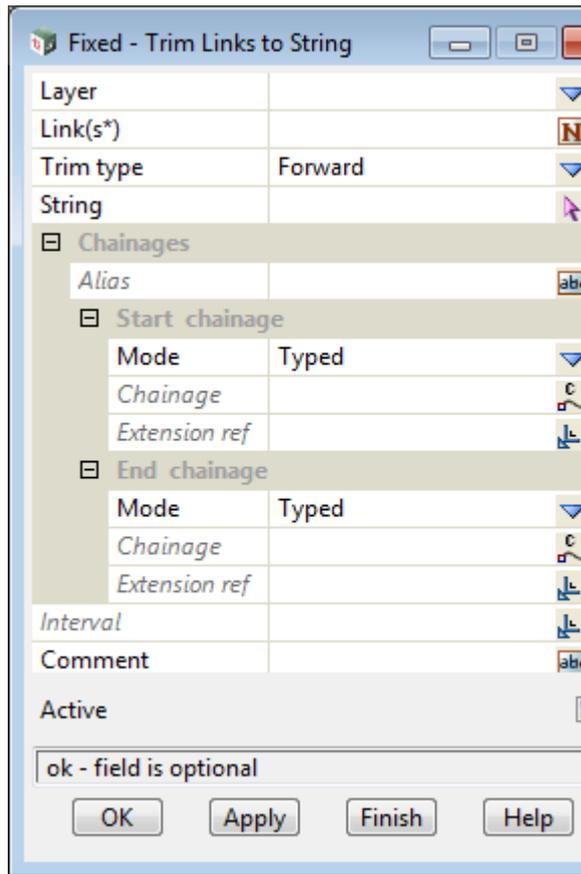


**Important Note:**

that this option only trims the strings that are created from the links. It does NOT trim the sections. So it is a process that is performed **after** the sections and strings are first generated and it is only performed on the strings.



Selecting Trim links to string brings up the Fixed - Trim Links to String panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>Layer of link to trim.</i>	layer box		available Layers
<b>Link(s)</b> <i>name of the links to trim against a string.</i>	name box		select name menu
<b>Trim type</b> <i>type of the trim for the links</i>  <i>if <b>forward</b>, there is a <b>String</b> select field and a string is user selected. All the links are trimmed so that they stop after the selected string.</i>  <i>If <b>between</b>, there is a <b>Start string</b> and <b>End string</b> select field and both strings are user selected. All the links are trimmed so that they stop between <b>Start string</b> and <b>End string</b>.</i>  <i>If <b>backward</b>, there is a <b>String</b> select field and a string is user selected. All the links are trimmed so that they only start at the selected string.</i>	choice box		forward, between, backward

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

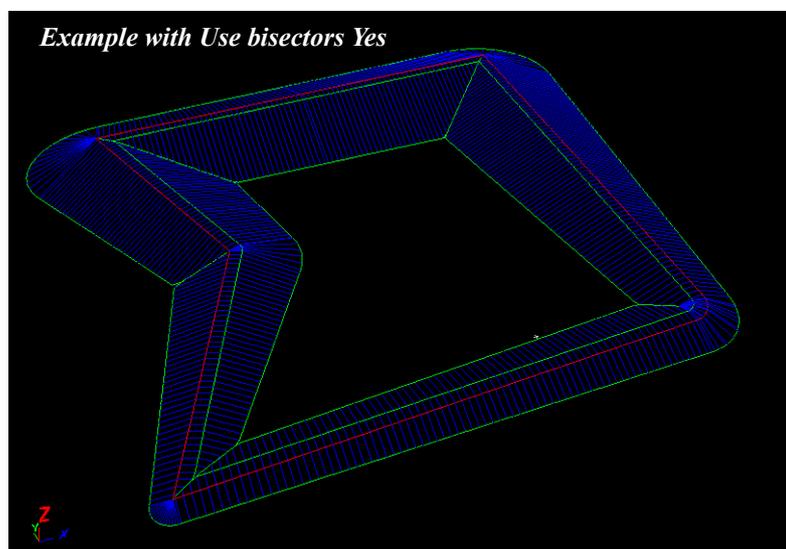
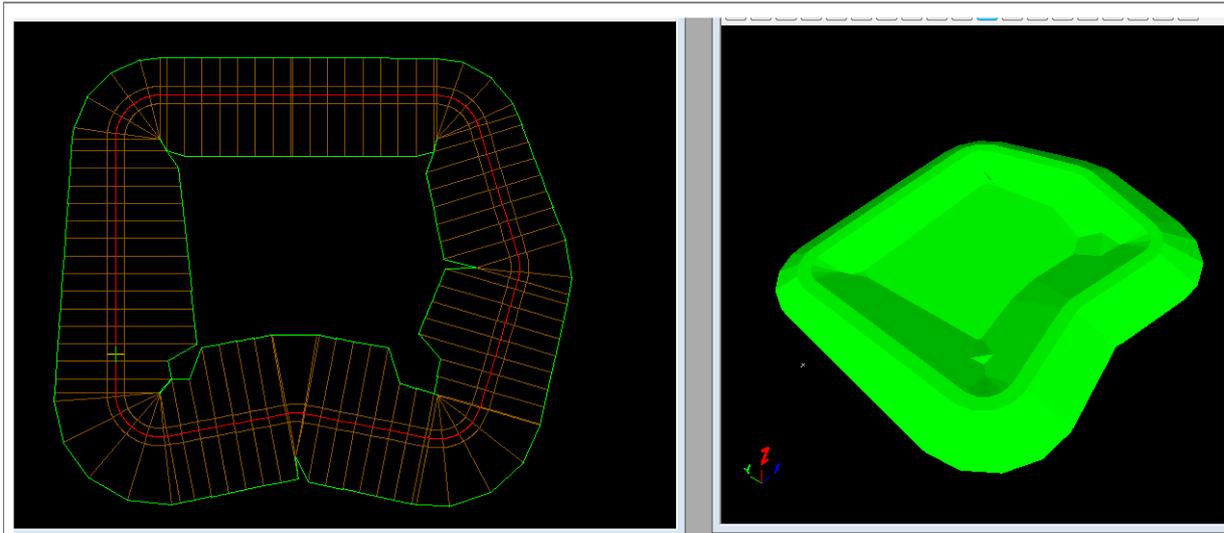
### 24.12.4.3.5 Trim Overlapping Sections

The **Trim Overlapping Sections** option takes a Layer and modifies the internally calculated sections so that the sections do not overlay each other.

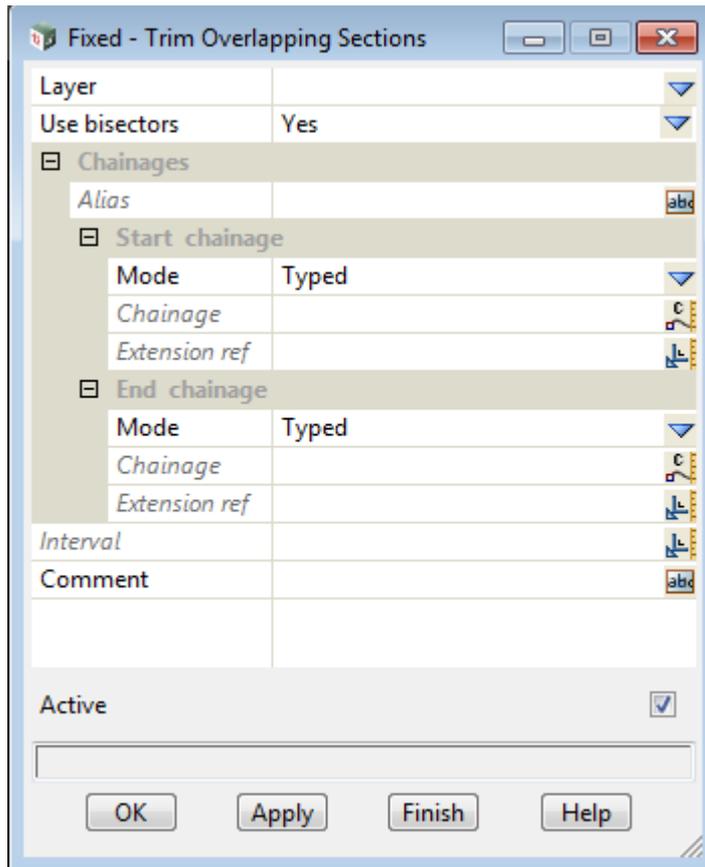
Any strings that are generated for that Layer will come from those modified internal sections and so will also not cross over themselves.

So any triangulation using those strings should not need any further cleanups to produce a fairly accurate tin. Certainly the tin should be accurate enough for any earthworks.

For the Design Layer, both sections and strings can be created and using the **Trim Overlapping Sections** option will produce results such as the example below.



Selecting Trim overlapping sections brings up the Fixed - Trim Overlapping Sections panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the Layer to modify the internally calculated sections so that the sections do not overlay each other. Any strings that are generated for that Layer after this process will come from the modified internal sections and so will also not cross over themselves.</i>	layer box		available Layers
<b>Use bisectors</b> <i>if Yes, then when there is no curve on the vertex between two straight segments then a bisector is created at the vertex where the angle is less than 180 degrees and used to help trim overlapping sections and strings. If No, then no extra bisectors are used.</i>	choice box	Yes	Yes/No

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.4 Left/Right Modifiers => Create =>Fixed =>Link

The **Fixed =>Link** option can modify the width, height, xfall or slope of any fixed link.

For **Fixed =>Link**, it doesn't matter which of the two values from width, height or xfall were used to defined the fixed link, the **Link** option allows for any combination to specify how to **modify** one value of the link whilst **holding** another value of the link to be what it is currently defined as by the fixed link.



So the **Link** option replaces the V10 **Fixed** options

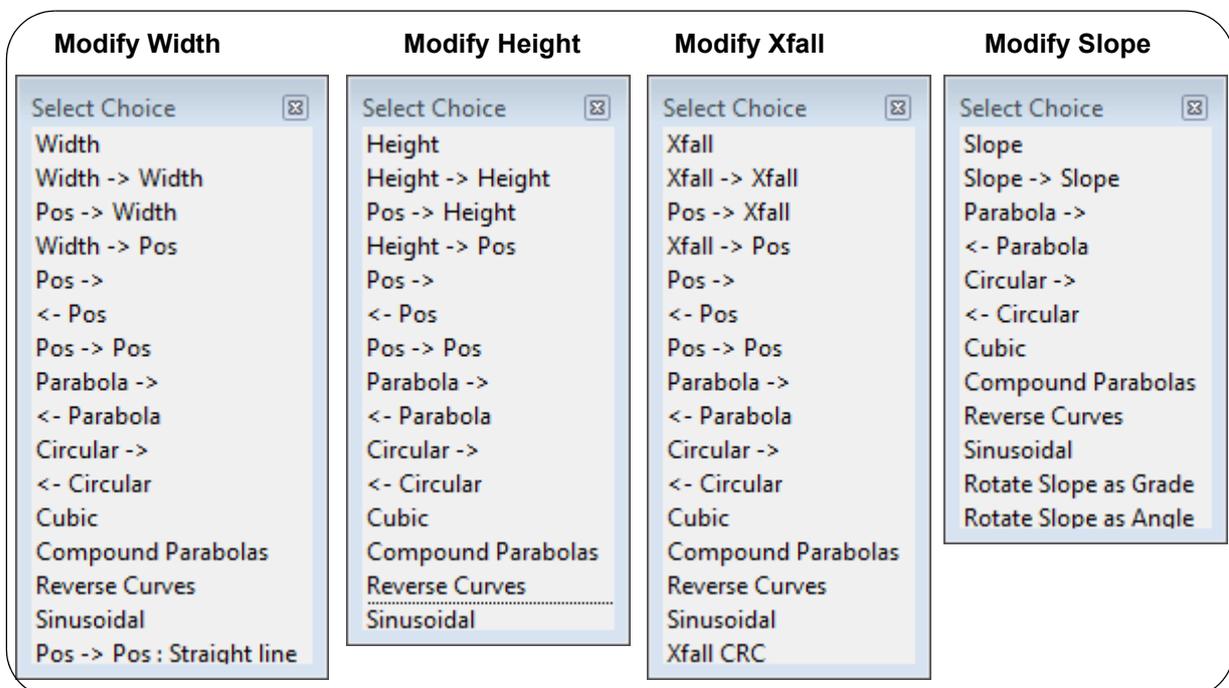
**Fixed =>Width**

**Fixed =>Height**

**Fixed =>Xfall**

**Fixed =>Xfall CRC**

Another difference is that there are new methods for modifying the values of width, height, xfall or slope.



For example, over the chainage range, the **width** of the link can follow a parabolic, circular curve or cubic shape.

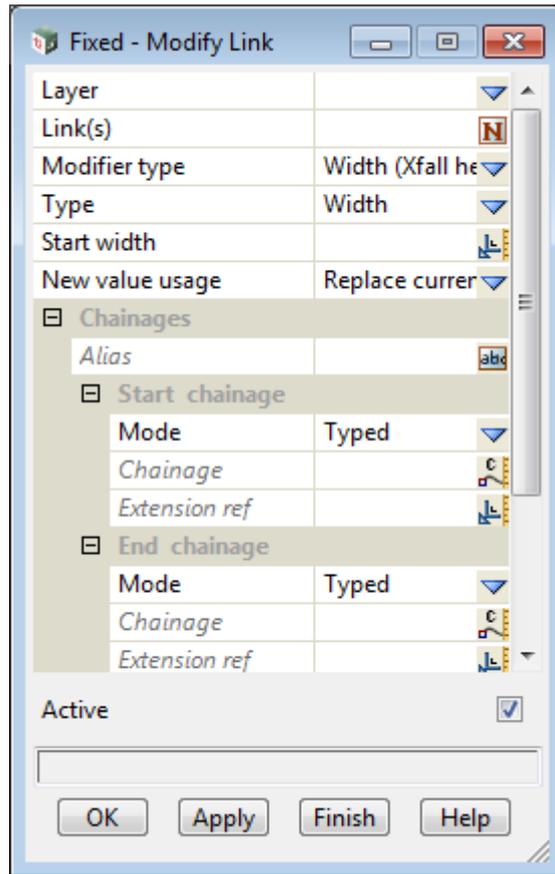
**Important Note:**

There is an important difference between Modify Xfall and Modify Slope.

**Modify Xfall** linearly interpolates the xfall between the Start Chainage and the End Chainage where xfall is the percentage of the proportion of metres vertically to metres horizontally.

**Modify Slope** linearly interpolates the slope between the Start Chainage and the End Chainage where slope is the ratio between one unit vertically to a number of units horizontally (1 in).

Another change is that **Absolute** in the *Modifier Fixed* => *Link* command has been replaced by a choice box **New value usage** and it has been moved to just under **Modifier type**.

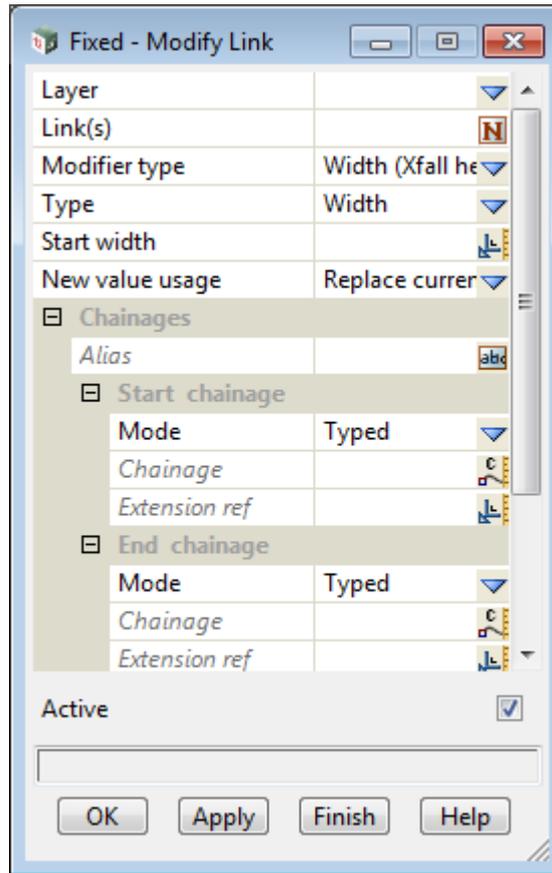


**New value usage** choice box Add to the current value of the link,  
Replace the current value of the link

*if Add to the current value of the link, the value of the parameter at each chainage is added to the current value for the parameter. This is the replacement for Absolute not ticked.*

*if Replace the current value of the link, the value of the parameter at each chainage replaces the current value for the parameter. This is the replacement for Absolute ticked.*

Selecting **Link** brings up the **Fixed - Modify Link** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b>	layer box	Design	available layers

*name of the layer that the link to be modified is in.*

<b>Link name</b>	name box		names.4d pop-up
------------------	----------	--	-----------------

*the name of the link in the Layer to modify the width/height/xfall/slope using the method given in **Modifier type**, between the chainages given by **Start chainage** and **End chainage**.*

*This is the current link.*

<b>Modifier type</b>	choice box		
----------------------	------------	--	--

modify Width and take Height from current link  
 modify Width and take Xfall from current link  
 modify Height and take Width from current link  
 modify Height and take Xfall from current link  
 modify Xfall and take Width from current link  
 modify Xfall and take Height from current link  
 modify Slope and take Width from current link  
 modify Slope and take Height from current link



*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).*

*The **Modify** width/height/xfall/slope is the part of the current link that is modified from its current value*

by the method given by **Modifier type**.

How the width/height/xfall/slope is modified depends on the choice made in the **Type** field described below.

The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.

**New value usage**                      choice box                      Add to the current value of the link,  
Replace the current value of the link

if **Add to the current value of the link**, the value of the parameter width/height/xfall/slope at each chainage is **added** to the current value for the parameter. This is the replacement for **Absolute not ticked**.

if **Replace the current value of the link**, the value of the parameter width/height/xfall/slope at each chainage **replaces** the current value for the parameter. This is the replacement for **Absolute ticked**.

**Type**    choice box

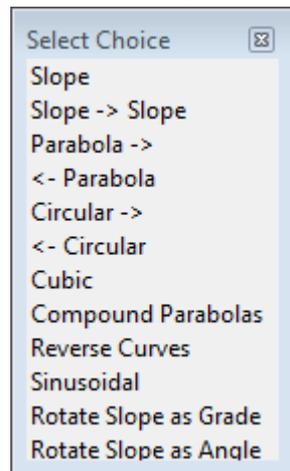
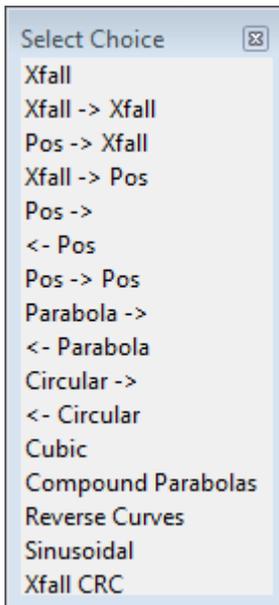
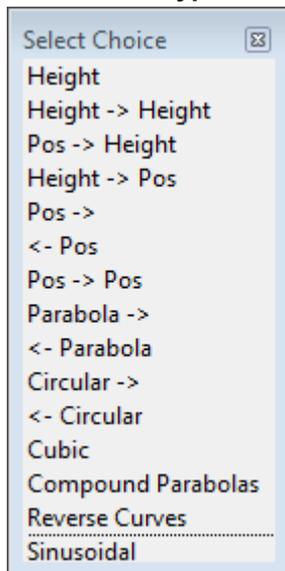
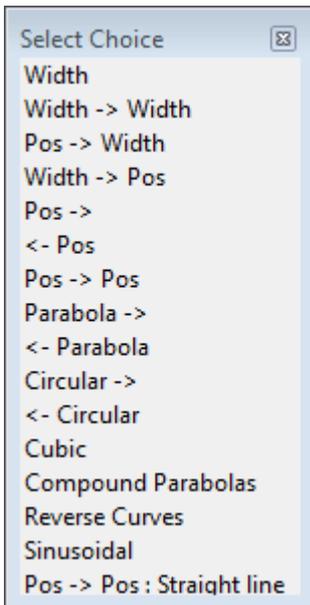
the choices for **Type** depends on the choice selected for **Modifier type**.

When **Modify type** is **Modify Width** the choices for **Type** are

When **Modify type** is **Modify Height** the choices for **Type** are

When **Modify type** is **Modify Xfall** the choices for **Type** are

When **Modify type** is **Modify Slope** the choices for **Type** are



For a description of the choices of **Type** when **Modifier type** is **Modify Width**, see [24.12.4.4.1 Type for Modifier Types "Modify Width, Hold Height/Xfall"](#)

For a description of the choices of **Type** when **Modifier type** is **Modify Height**, see [24.12.4.4.2 Type for Modifier Types "Modify Height, Hold Width/Xfall"](#)

For a description of the choices of **Type** when **Modifier type** is **Modify Xfall**, see [24.12.4.4.3 Type for Modifier Types "Modify Xfall, Hold Width/Height"](#)

For a description of the choices of **Type** when **Modifier type** is **Modify Slope**, see [24.12.4.4.4 Type for Modifier Types "Modify Slope, Hold Width/Height"](#)

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

### 24.12.4.4.1 Type for Modifier Types "Modify Width, Hold Height/Xfall"

Modify Width	
Select Choice <input type="checkbox"/>	
Width	use a given width for the chainage range
Width -> Width	interpolate between a given start width and a given end width
Pos -> Width	interpolate between the current start width and a given end width
Width -> Pos	interpolate between a given start width and the current end width
Pos ->	use the current start width for the chainage range
<- Pos	use the current start width for the chainage range
Pos -> Pos	interpolate between the current start width and the current end width
Parabola ->	use a parabola from the current start width
<- Parabola	use a parabola that goes through the current end width
Circular ->	use an arc from the current start width
<- Circular	use an arc that goes through the current end width
Cubic	use a cubic between the current start and end widths
Compound Parabolas	use a compound parabola between the current start and end widths
Reverse Curves	use reverse curves between the current start and end widths
Sinusoidal	use a sinusoidal curve between the current start and end widths
Pos -> Pos : Straight line	use a straight line between the current start and end widths

**For Type**

- [24.12.4.4.1.1 Width](#)
- [24.12.4.4.1.2 Width -> Width](#)
- [24.12.4.4.1.3 Pos -> Width](#)
- [24.12.4.4.1.4 Width -> Pos](#)
- [24.12.4.4.1.5 Pos ->](#)
- [24.12.4.4.1.6 <- Pos](#)
- [24.12.4.4.1.7 Pos-> Pos:](#)
- [24.12.4.4.1.8 Parabola ->](#)
- [24.12.4.4.1.9 <- Parabola](#)
- [24.12.4.4.1.10 Circular->](#)
- [24.12.4.4.1.11 <-Circular](#)
- [24.12.4.4.1.12 Cubic](#)
- [24.12.4.4.1.13 Pos-> Pos: Straight line](#)

### 24.12.4.4.1.1 Width

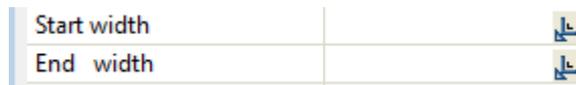
#### Keep a Given Width



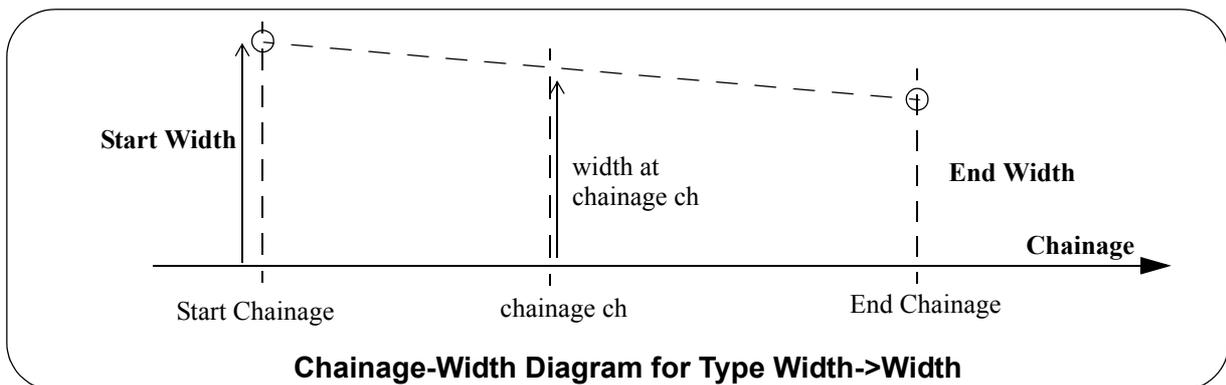
At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is that given in the **Width** panel field. That is, the width has the constant value of **Width**.

### 24.12.4.4.1.2 Width -> Width

#### Interpolate Between a Given Start and End Width



At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is linearly interpolated by chainage between the given **Start width** and the **End width**.



### 24.12.4.4.1.3 Pos -> Width

#### Interpolate Between Existing Start Width & Given End width



At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is linearly interpolated by chainage between the width at the **Start chainage** and the given width **Width** at the **End chainage**.

This allows you to use the width at the **End chainage** without knowing what the actual value of the width is.

#### 24.12.4.4.1.4 Width -> Pos

##### Interpolate Between a Given Start Width & an Existing End Width



At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is linearly interpolated by chainage between the given width **Width** at the **Start chainage** and the width at the **End chainage**.

This allows you to use the width at the **Start chainage** without knowing what the actual value of the width is.

#### 24.12.4.4.1.5 Pos ->

##### Keep the Same Width as at the Start Chainage

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is that at the **Start chainage**. That is, the width has the constant value of what it is at the **Start chainage**.

This allows you to use the width at the **Start chainage** without knowing what the actual value of the width is.

#### 24.12.4.4.1.6 <- Pos

##### Keep the Same Width as at the End chainage

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is that at the **End chainage**. That is, the width has the constant value of what it is at the **End chainage**.

This allows you to use the width at the **End chainage** without knowing what the actual value of the width is.

#### 24.12.4.4.1.7 Pos-> Pos:

##### Interpolate Between Existing Widths at Start & End Chainages

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is linearly interpolated by chainage between the width at the **Start chainage** and the width at the **End chainage**.

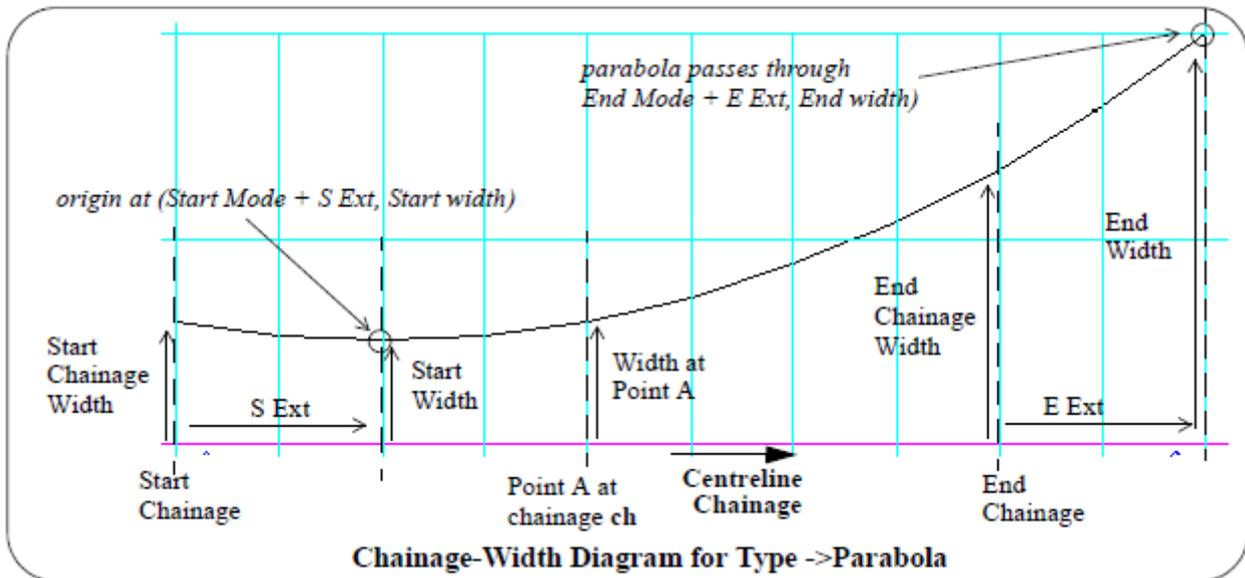
This allows you to use the widths at the **Start chainage** and the **End chainage** without knowing what the actual value of the widths are.

24.12.4.1.8 Parabola ->

Calculation of Width for Type Parabola ->

In the chainage-width diagram, the **Start width** is the width at chainage **Start mode plus S Ext**, and **End width** is the width at chainage **End mode plus E Ext**.

Between the start position (**Start mode + S Ext, Start width**), and the end position (**End mode + E Ext, End width**), the width varies as a **parabola  $a \cdot X^2$**  with its origin at (**Start mode + S Ext, Start width**) and going through (**End mode + E Ext, End width**).

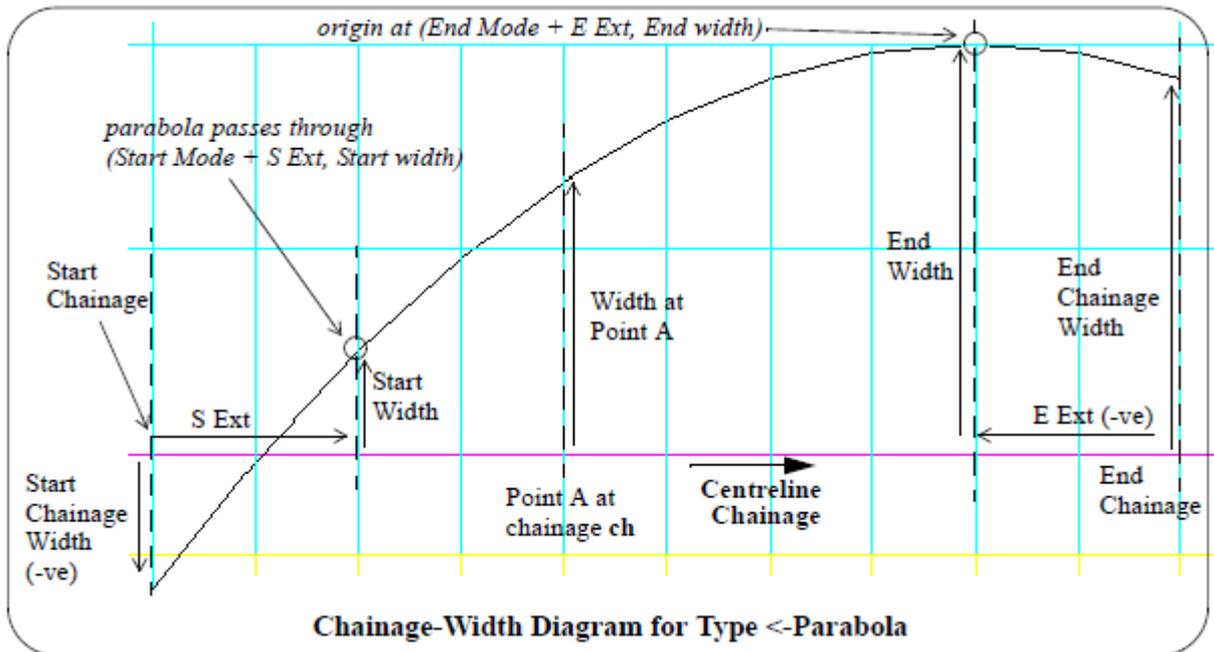


### 24.12.4.4.1.9 <- Parabola

#### Calculation of Width for Type <- Parabola

In the chainage-width diagram, the **Start width** is the width at chainage **Start mode plus S Ext**, and **End width** is the width at chainage **End mode plus E Ext**.

Between the start position (**Start mode + S Ext, Start width**), and the end position (**End mode + E Ext, End width**), the width varies as a **parabola**  $a*X*X$  with its origin at (**End mode + E Ext, End width**) and going through (**Start mode + S Ext, Start width**).

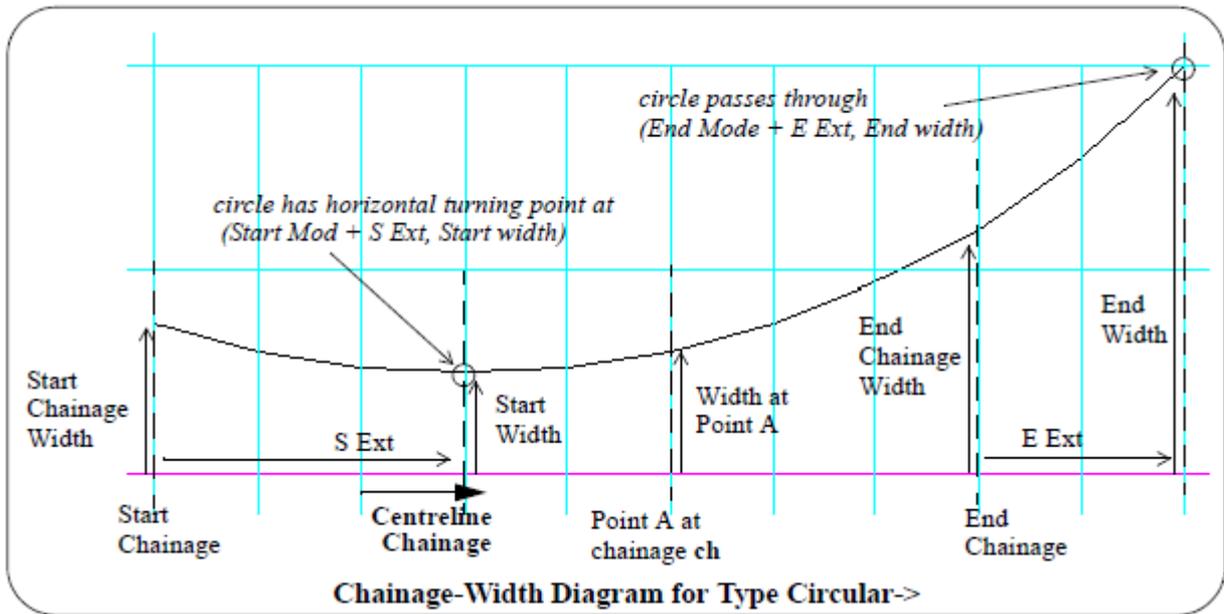


24.12.4.4.1.10 Circular->

Calculation of Width for Type Circular->

In the chainage-width diagram, the **Start width** is the width at chainage **Start mode plus S Ext**, and **End width** is the width at chainage **End mode plus E Ext**.

Between the start position (**Start mode + S Ext, Start width**), and the end position (**End mode + E Ext, End width**), the width varies as a **circle** with its horizontal turning point at (**Start mode + S Ext, Start width**) and going through (**End mode + E Ext, End width**).

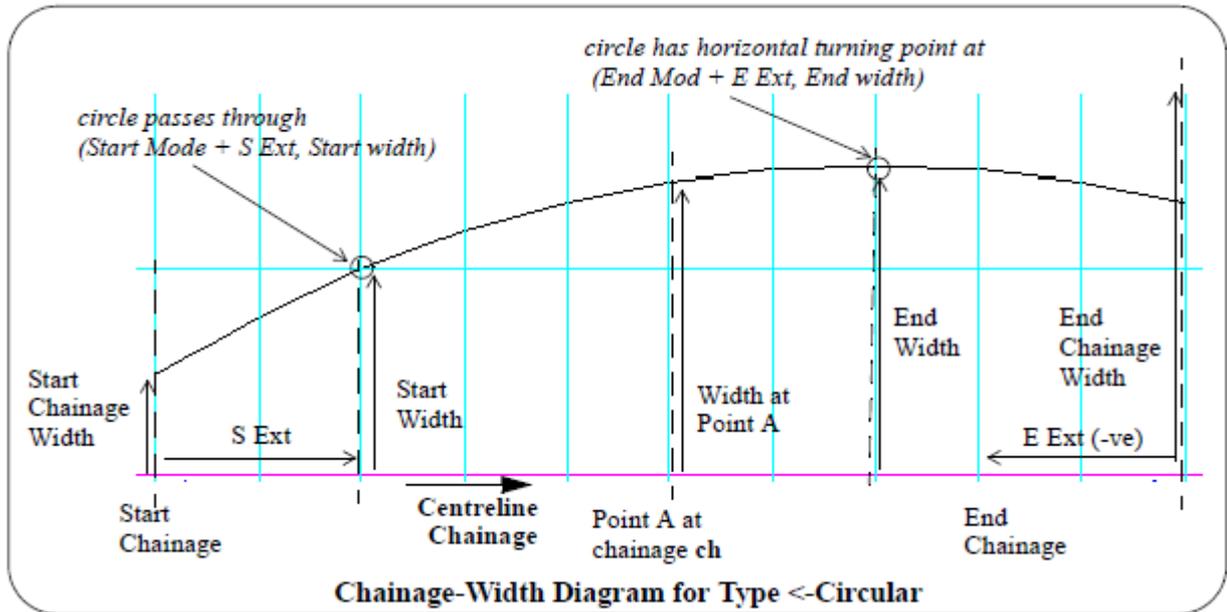


### 24.12.4.4.1.11 <-Circular

#### Calculation of Width for Type <-Circular

In the chainage-width diagram, the **Start width** is the width at chainage **Start mode plus S Ext**, and **End width** is the width at chainage **End mode plus E Ext**.

Between the start position (**Start mode + S Ext, Start width**), and the end position (**End mode + E Ext, End width**), the width varies as a **circle** with its horizontal turning point at (**End mode + E Ext, End width**) and going through (**Start mode + S Ext, Start width**).

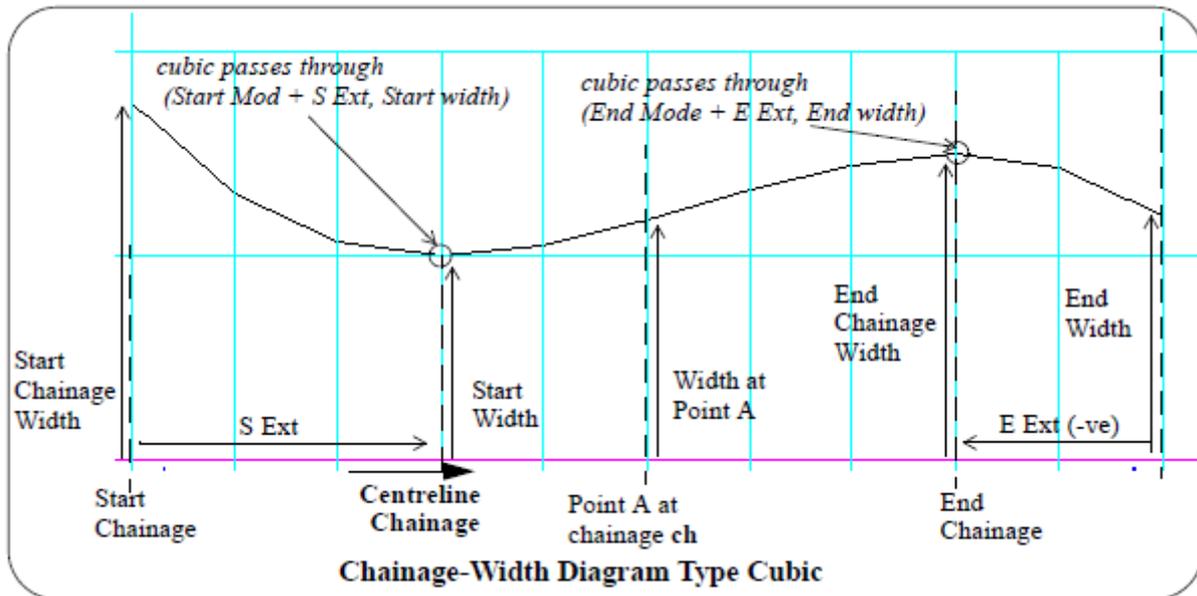


### 24.12.4.4.1.12 Cubic

#### Calculation of Width for Type Cubic

In the chainage-width diagram, the **Start width** is the width at chainage **Start mode plus S Ext**, and **End width** is the width at chainage **End mode plus E Ext**.

Between the start position (**Start mode + S Ext, Start width**), and the end position (**End mode + E Ext, End width**), the width varies as a **cubic** going through (**Start mode + S Ext, Start width**) and (**End mode + E Ext, End width**).



### 24.12.4.4.1.13 Pos-> Pos: Straight line

#### Join Between Existing Widths at Start & End Chainages with a Straight Plan Line

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the width is calculated so that it is on a straight line in plan between the width at the **Start chainage** and the width at the **End chainage**.

This allows you to join the widths at the **Start chainage** and the **End chainage** by a plan straight line without knowing what the actual value of the widths are.

## 24.12.4.4.2 Type for Modifier Types "Modify Height, Hold Width/Xfall"

### Modify Height

Select Choice	
Height	use a given height for chainage range
Height -> Height	interpolate between a given start height and a given end height
Pos -> Height	interpolate between the current start height and a given end height
Height -> Pos	interpolate between a given start height and the current end height
Pos ->	keep the current start height for the chainage range
<- Pos	use the current end height for the chainage range
Pos -> Pos	interpolate between the current start height and the current end height
Parabola ->	use a parabola from the current start height
<- Parabola	use a parabola that goes through the current end height
Circular ->	use an arc from the current start height
<- Circular	use an arc that goes through the current end height
Cubic	use a cubic that goes through the current start and end heights
Compound Parabolas	use a compound parabola between the current start and end heights
Reverse Curves	use reverse curves between the current start and end heights
Sinusoidal	use a sinusoidal curve between the current start and end heights

### For Type

[24.12.4.4.2.1 Height](#)

[24.12.4.4.2.2 Height -> Height](#)

[24.12.4.4.2.3 Pos -> Height](#)

[24.12.4.4.2.4 Height -> Pos](#)

[24.12.4.4.2.5 Pos ->](#)

[24.12.4.4.2.6 <- Pos](#)

[24.12.4.4.2.7 Pos-> Pos:](#)

[24.12.4.4.2.8 Parabola ->](#)

[24.12.4.4.2.9 <- Parabola](#)

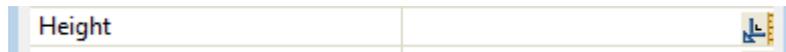
[24.12.4.4.2.10 Circular->](#)

[24.12.4.4.2.11 <-Circular](#)

[24.12.4.4.2.12 Cubic](#)

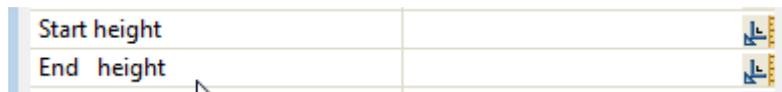
[24.12.4.4.2.12 Cubic](#)

**24.12.4.4.2.1 Height**  
**Keep a Given Height**

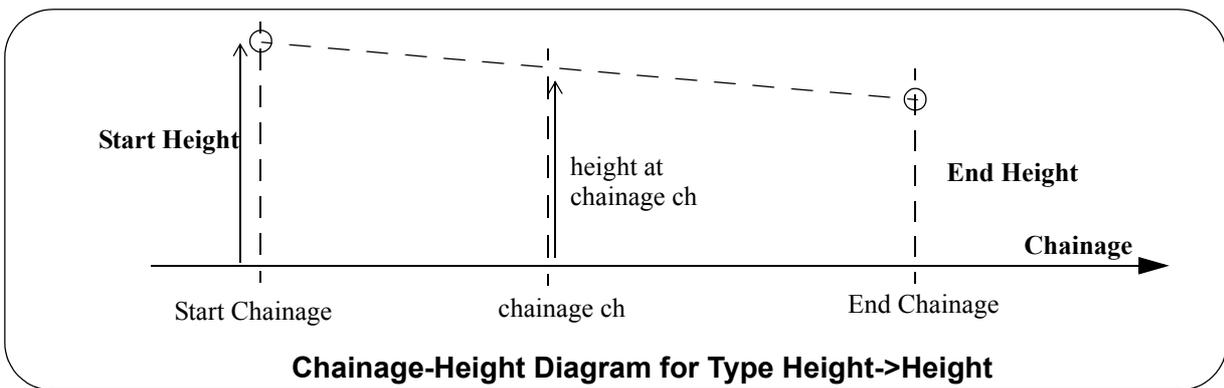


At each chainage **ch** between the **Start chainage** and **End chainage**, the width is that given in the **Height** panel field. That is, the height has the constant value of **Height**.

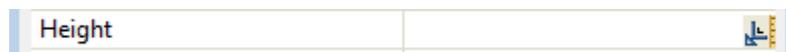
**24.12.4.4.2.2 Height -> Height**  
**Interpolate Between a Given Start and End Height**



At each chainage **ch** between the **Start chainage** and **End chainage**, the height is linearly interpolated by chainage between the given **Start height** and the **End height**.



**24.12.4.4.2.3 Pos -> Height**  
**Interpolate Between Existing Start Height & Given End height**

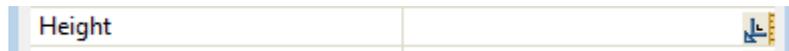


At each chainage **ch** between the **Start chainage** and **End chainage**, the height is linearly interpolated by chainage between the height at the **Start chainage** and the given height **Height** at the **End chainage**.

This allows you to use the height at the **End chainage** without knowing what the actual value of the height is.

#### 24.12.4.4.2.4 Height -> Pos

##### Interpolate Between a Given Start Height & an Existing End Height



At each chainage  $ch$  between the **Start chainage** and **End chainage**, the height is linearly interpolated by chainage between the given height **Height** at the **Start chainage** and the height at the **End chainage**.

This allows you to use the height at the **Start chainage** without knowing what the actual value of the height is.

#### 24.12.4.4.2.5 Pos ->

##### Keep the Same Height as at the Start Chainage

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the height is that at the **Start chainage**. That is, the height has the constant value of what it is at the **Start chainage**.

This allows you to use the height at the **Start chainage** without knowing what the actual value of the height is.

#### 24.12.4.4.2.6 <- Pos

##### Keep the Same Height as at the End chainage

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the height is that at the **End chainage**. That is, the height has the constant value of what it is at the **End chainage**.

This allows you to use the height at the **End chainage** without knowing what the actual value of the height is.

#### 24.12.4.4.2.7 Pos-> Pos:

##### Interpolate Between Existing Heights at Start & End Chainages

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the height is linearly interpolated by chainage between the height at the **Start chainage** and the height at the **End chainage**.

This allows you to use the heights at the **Start chainage** and the **End chainage** without knowing what the actual value of the heights are.

#### 24.12.4.4.2.8 Parabola ->

##### Calculation of Height for Type Parabola ->

*The calculations are the same as the case for Width except Height is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.8 Parabola ->](#).*

#### **24.12.4.4.2.9 <- Parabola**

##### **Calculation of Height for Type <- Parabola**

*The calculations are the same as the case for Width except Height is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.9 <- Parabola](#).*

#### **24.12.4.4.2.10 Circular->**

##### **Calculation of Height for Type Circular->**

*The calculations are the same as the case for Width except Height is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.10 Circular->](#).*

#### **24.12.4.4.2.11 <-Circular**

##### **Calculation of Height for Type <-Circular**

*The calculations are the same as the case for Width except Height is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.11 <-Circular](#).*

#### **24.12.4.4.2.12 Cubic**

##### **Calculation of Height for Type Cubic**

*The calculations are the same as the case for Width except Height is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.12 Cubic](#).*

### 24.12.4.4.3 Type for Modifier Types "Modify Xfall, Hold Width/Height"

Modify Xfall	
Select Choice <input type="checkbox"/>	
Xfall	use a given xfall for chainage range
Xfall -> Xfall	interpolate between a given start xfall and a given end xfall
Pos -> Xfall	interpolate between the current start xfall and a given end xfall
Xfall -> Pos	interpolate between a given start xfall and the current end xfall
Pos ->	keep the current start xfall for the chainage range
<- Pos	use the current end xfall for the chainage range
Pos -> Pos	interpolate between the current start xfall and the current end xfall
Parabola ->	use a parabola from the current start xfall
<- Parabola	use a parabola that goes through the current end xfall
Circular ->	use an arc from the current start xfall
<- Circular	use an arc that goes through the current end xfall
Cubic	use a cubic between the current xfall and end xfall
Compound Parabolas	use a compound parabola between the current xfall and end xfall
Reverse Curves	use reverse curves between the current start and end xfalls
Sinusoidal	use a sinusoidal curve between the current start and end xfalls
Xfall CRC	

#### For Type

[24.12.4.4.3.1 Xfall](#)

[24.12.4.4.3.2 Xfall -> Xfall](#)

[24.12.4.4.3.3 Pos -> Xfall](#)

[24.12.4.4.3.4 Xfall -> Pos](#)

[24.12.4.4.3.5 Pos ->](#)

[24.12.4.4.3.6 <- Pos](#)

[24.12.4.4.3.7 Pos-> Pos:](#)

[24.12.4.4.3.8 Parabola ->](#)

[24.12.4.4.3.9 <- Parabola](#)

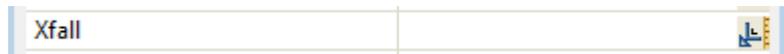
[24.12.4.4.3.10 Circular->](#)

[24.12.4.4.3.11 <-Circular](#)

[24.12.4.4.3.12 Cubic](#)

### 24.12.4.4.3.1 Xfall

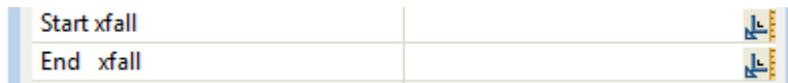
#### Keep a Given Xfall



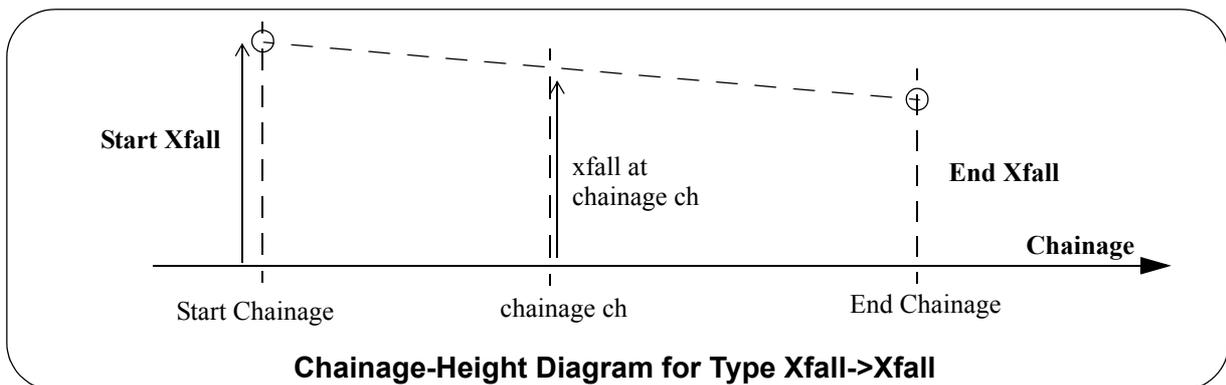
At each chainage *ch* between the **Start chainage** and **End chainage**, the width is that given in the **Xfall** panel field. That is, the Xfall has the constant value of **Xfall**.

### 24.12.4.4.3.2 Xfall -> Xfall

#### Interpolate Between a Given Start and End Xfall

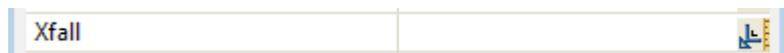


At each chainage *ch* between the **Start chainage** and **End chainage**, the Xfall is linearly interpolated by chainage between the given **Start Xfall** and the **End Xfall**.



### 24.12.4.4.3.3 Pos -> Xfall

#### Interpolate Between Existing Start Xfall & Given End Xfall

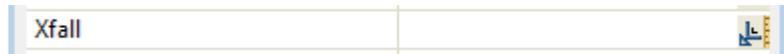


At each chainage *ch* between the **Start chainage** and **End chainage**, the Xfall is linearly interpolated by chainage between the Xfall at the **Start chainage** and the given Xfall **Xfall** at the **End chainage**.

This allows you to use the Xfall at the **End chainage** without knowing what the actual value of the Xfall is.

#### 24.12.4.4.3.4 Xfall -> Pos

##### Interpolate Between a Given Start Xfall & an Existing End Xfall



At each chainage  $ch$  between the **Start chainage** and **End chainage**, the Xfall is linearly interpolated by chainage between the given Xfall **Xfall** at the **Start chainage** and the Xfall at the **End chainage**.

This allows you to use the Xfall at the **Start chainage** without knowing what the actual value of the Xfall is.

#### 24.12.4.4.3.5 Pos ->

##### Keep the Same Xfall as at the Start Chainage

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the Xfall is that at the **Start chainage**. That is, the Xfall has the constant value of what it is at the **Start chainage**.

This allows you to use the Xfall at the **Start chainage** without knowing what the actual value of the Xfall is.

#### 24.12.4.4.3.6 <- Pos

##### Keep the Same Xfall as at the End chainage

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the Xfall is that at the **End chainage**. That is, the Xfall has the constant value of what it is at the **End chainage**.

This allows you to use the Xfall at the **End chainage** without knowing what the actual value of the Xfall is.

#### 24.12.4.4.3.7 Pos-> Pos:

##### Interpolate Between Existing Xfalls at Start & End Chainages

At each chainage  $ch$  between the **Start chainage** and **End chainage**, the Xfall is linearly interpolated by chainage between the Xfall at the **Start chainage** and the Xfall at the **End chainage**.

This allows you to use the Xfalls at the **Start chainage** and the **End chainage** without knowing what the actual value of the heights are.

#### 24.12.4.4.3.8 Parabola ->

##### Calculation of Xfall for Type Parabola ->

*The calculations are the same as the case for Width except Xfall is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.8 Parabola ->](#).*

#### 24.12.4.4.3.9 <- Parabola

##### Calculation of Xfall for Type <- Parabola

*The calculations are the same as the case for Width except Xfall is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.9 <- Parabola](#).*

#### **24.12.4.4.3.10 Circular->**

##### **Calculation of Xfall for Type Circular->**

*The calculations are the same as the case for Width except Xfall is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.10 Circular->](#).*

#### **24.12.4.4.3.11 <-Circular**

##### **Calculation of Xfall for Type <-Circular**

*The calculations are the same as the case for Width except Xfall is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.11 <-Circular](#).*

#### **24.12.4.4.3.12 Cubic**

##### **Calculation of Xfall for Type Cubic**

*The calculations are the same as the case for Width except Height is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.12 Cubic](#).*

## 24.12.4.4.4 Type for Modifier Types "Modify Slope, Hold Width/Height"

Modify Slope	
Select Choice	
Slope	use a given slope for chainage range
Slope -> Slope	interpolate between a given start slope and a given end slope
Parabola ->	use a parabola from the current start slope
<- Parabola	use a parabola that goes through the current end slope
Circular ->	use an arc from the current start slope
<- Circular	use an arc that goes through the current end slope
Cubic	use a cubic between the current start and end slope
Compound Parabolas	use a compound parabola between the current start and end slope
Reverse Curves	use reverse curves between the current start and end slopes
Sinusoidal	use a sinusoidal curve between the current start and end slopes
Rotate Slope as Grade	
Rotate Slope as Angle	

### For Type

[24.12.4.4.4.1 Slope](#)

[24.12.4.4.4.2 Slope -> Slope](#)

[24.12.4.4.4.3 Parabola ->](#)

[24.12.4.4.4.4 <- Parabola](#)

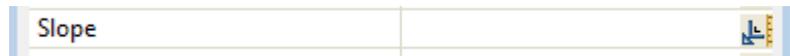
[24.12.4.4.4.5 Circular->](#)

[24.12.4.4.4.6 <-Circular](#)

[24.12.4.4.4.7 Cubic](#)

#### 24.12.4.4.1 Slope

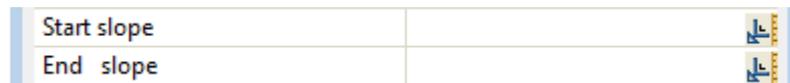
##### Keep a Given Slope



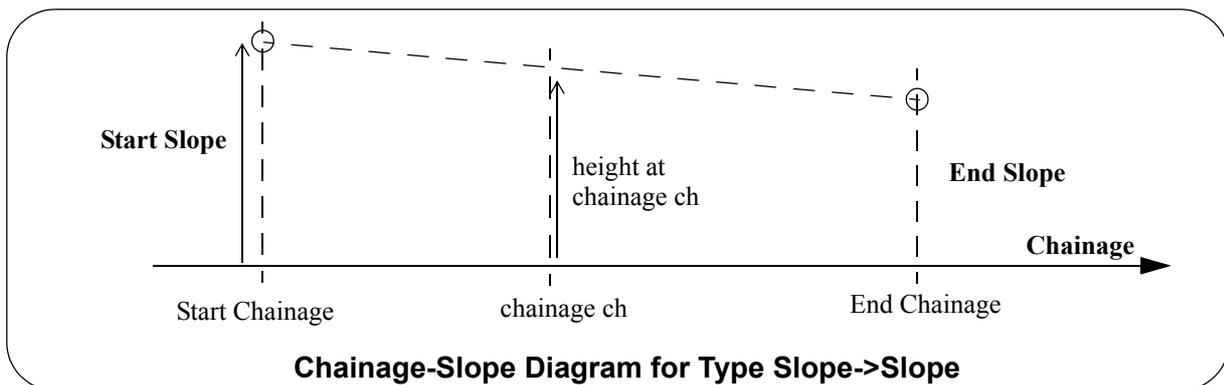
At each chainage *ch* between the **Start chainage** and **End chainage**, the width is that given in the **Slope** panel field. That is, the slope has the constant value of **Slope**.

#### 24.12.4.4.2 Slope -> Slope

##### Interpolate Between a Given Start and End Slope



At each chainage *ch* between the **Start chainage** and **End chainage**, the slope is linearly interpolated by chainage between the given **Start slope** and the **End slope**.



#### 24.12.4.4.3 Parabola ->

##### Calculation of Slope for Type Parabola ->

The calculations are the same as the case for Width except Slope is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.8 Parabola ->](#).

#### 24.12.4.4.4 <- Parabola

##### Calculation of Slope for Type <- Parabola

The calculations are the same as the case for Width except Slope is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.9 <- Parabola](#).

#### 24.12.4.4.5 Circular->

##### Calculation of Slope for Type Circular->

The calculations are the same as the case for Width except Slope is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.10 Circular->](#).

#### **24.12.4.4.4.6 <-Circular**

##### **Calculation of Slope for Type <-Circular**

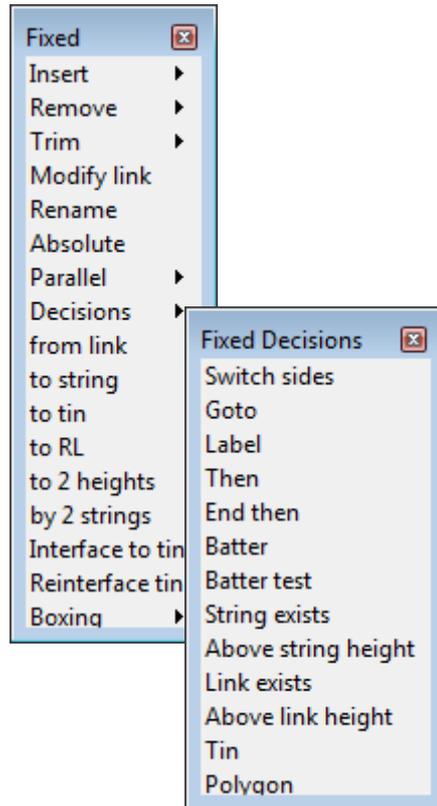
*The calculations are the same as the case for Width except Slope is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.11 <-Circular](#).*

#### **24.12.4.4.4.7 Cubic**

##### **Calculation of Slope for Type Cubic**

*The calculations are the same as the case for Width except Slope is substituted for Width in all the formulae and diagrams. See [24.12.4.4.1.12 Cubic](#).*

## 24.12.4.5 Left/Right Modifiers => Create =>Fixed =>Decisions



The Fixed =>Decisions bring the power of the **Decisions** from **Templates** to the **Left/Right Modifiers**.

See

[24.12.4.5.1 Left/Right Modifiers => Create =>Fixed =>Decisions =>Switch sides](#)

[24.12.4.5.2 Left/Right Modifiers => Create =>Fixed =>Decisions =>Goto](#)

[24.12.4.5.3 Left/Right Modifiers => Create =>Fixed =>Decisions =>Label](#)

[24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then](#)

[24.12.4.5.5 Left/Right Modifiers => Create =>Fixed =>Decisions =>End Then](#)

[24.12.4.5.6 Left/Right Modifiers => Create =>Fixed =>Decisions =>Batter](#)

[24.12.4.5.7 Left/Right Modifiers => Create =>Fixed =>Decisions =>Batter test](#)

[24.12.4.5.8 Left/Right Modifiers => Create =>Fixed =>Decisions =>String exists](#)

[24.12.4.5.9 Left/Right Modifiers => Create =>Fixed =>Decisions =>Above String Height](#)

[24.12.4.5.10 Left/Right Modifiers => Create =>Fixed =>Decisions =>Link Exists](#)

[24.12.4.5.11 Left/Right Modifiers => Create =>Fixed =>Decisions =>Above Link Height](#)

[24.12.4.5.12 Left/Right Modifiers => Create =>Fixed =>Decisions =>Tin](#)

[24.12.4.5.13 Left/Right Modifiers => Create =>Fixed =>Decisions =>Polygon](#)

### 24.12.4.5.1 Left/Right Modifiers => Create =>Fixed =>Decisions =>Switch sides

This option is new in V11.

The **Switch Sides** command transfers processing to the other side of the **Modifiers**.

That is, if a **Switch Sides** command is processed in a **Left MTF Modifiers** grid, then processing stops at that point in the **Left MTF Modifiers** grid, and processing is transferred to the **Right MTF Modifiers** grid. In the **Right MTF Modifiers** grid it from the row where the last **Switch Sides** command was executed **in the Right MTF Modifiers** grid, or if that hasn't occurred before, processing starts at the top row of the **Right MTF Modifiers** grid.

And similarly if a **Switch Sides** command is processed in a **Right MTF Modifiers** grid

So processing can switch a number of times between the two sides.

Selecting **Switch sides** brings up the panel **Modify Decision Switch Sides**

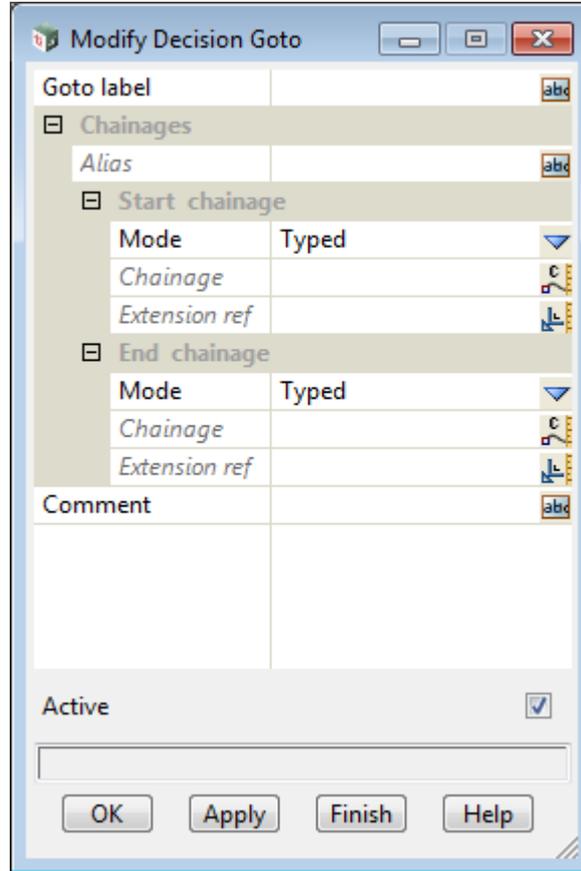
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.5.2 Left/Right Modifiers => Create =>Fixed =>Decisions =>Goto

This option is new in V11.

**Goto** will transfer control to the row of the **Left/Right MTF Modifiers** grid with the label **Goto label**.  
Selecting **Goto** brings up the panel **Modify Decision Goto**



**Goto label** text box

*transfer processing to the row of the Left/Right MTF Modifiers grid with this label.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.5.3 Left/Right Modifiers => Create =>Fixed =>Decisions =>Label

This option is new in V11.

The **Label** option creates a row in the **Left/Right MTF Modifiers** grid with the label **Label**.

Selecting **Label** brings up the panel **Modify Decision Label**

**Label** text box

*name of the label that goes with this row of the grid.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then

This option is new in V11.

The **Then** Command works in conjunction with an **End Then** command, and they both work with any one of the **Decision** commands that has a **test** and a **Goto Label** that is executed when the test succeeds (.e.g Batter Decision, String Exist, Tin).

The **Then** and the **End Then** must have the value for the **Label** field to be the same as the **GoTo Label** in a Test command.

If such an **Then-End Then** block exists for a Test command, then whenever the **GoTo Label** is executed for the Test command then processing is transferred to the **Then** command and continues from there.

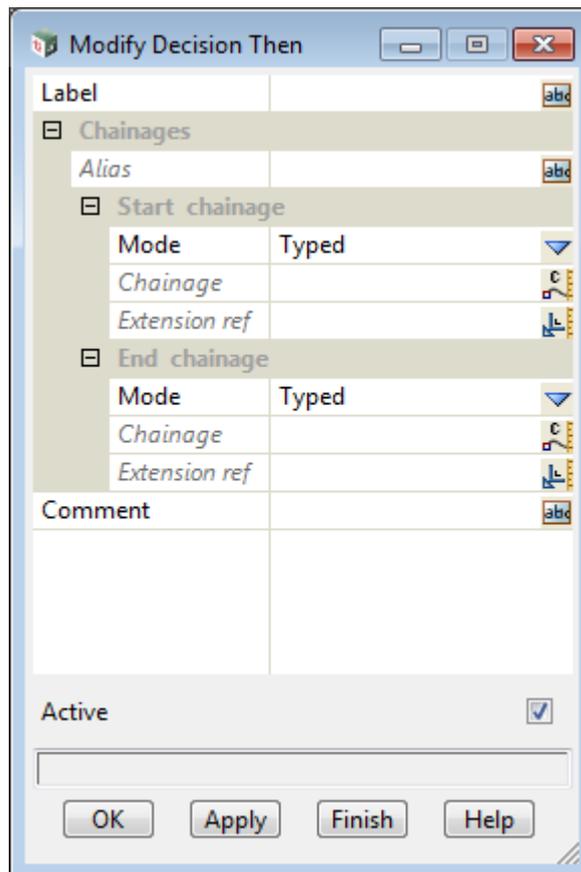
However if processing reaches a **Then** command by any mean other than from a Test command, then all the commands are ignored until the **End Then** command is reached.

So a **Then-End Then** block is ignored except by a Test command with a matching **GoTo Label**.

#### Notes

1. More than one Test command can use the same **Then-End Then** block.
2. Jumping into the middle of a **Then-End Then** block should not be done and will have unpredictable consequences.

Selecting **Then** brings up the **Modify Decision Then** panel



**Label** text box

*name of the label that is the same as its matching **End Then** command, and matches the **GoTo Label** of any Test command that wants to use the **Then-End Then** block.*

### 24.12.4.5.5 Left/Right Modifiers => Create =>Fixed =>Decisions =>End Then

This option is new in V11.

The **End Then** goes with a matching **Then** command. For a description of how the **Then-End Then** commands are used, see [24.12.4.5.5 Left/Right Modifiers => Create =>Fixed =>Decisions =>End Then](#).

Selecting **End then** brings up the panel **Modify Decision End Then**

**Label** text box

*name of the label that is the same as its matching **Then** command, and matches the **Goto Label** of any **Test** command that wants to use the **Then-End Then** block.*

### 24.12.4.5.6 Left/Right Modifiers => Create =>Fixed =>Decisions =>Batter

This option is new in V11.

Selecting **Batter** brings up the **Modify Decision Batter** panel which is used to construct a link which stops if it comes within

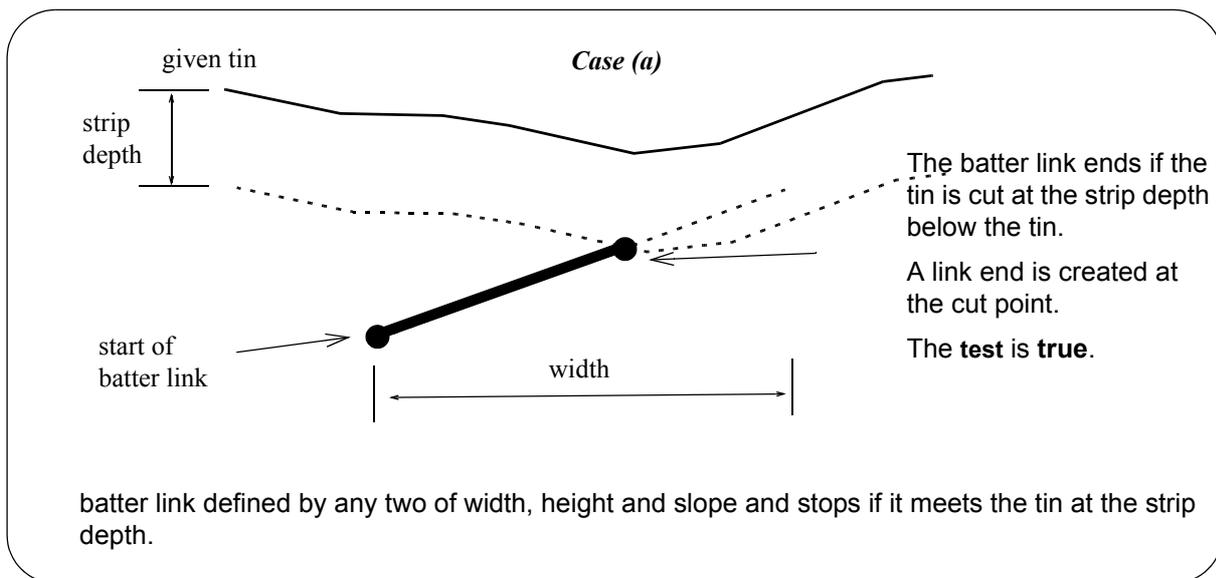
- (a) a **Strip** depth of a **Tin**
- (b) an **Offset** distance from a **Tin**.
- (c) a **Strip** depth of a **Tin** calculated at a given **Offset** from the link.

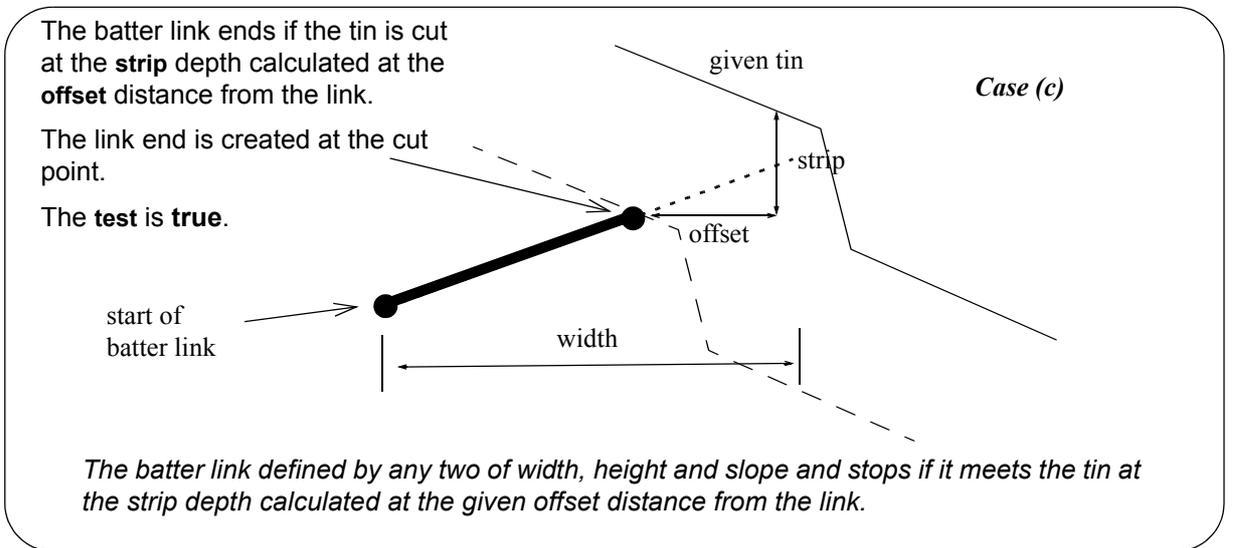
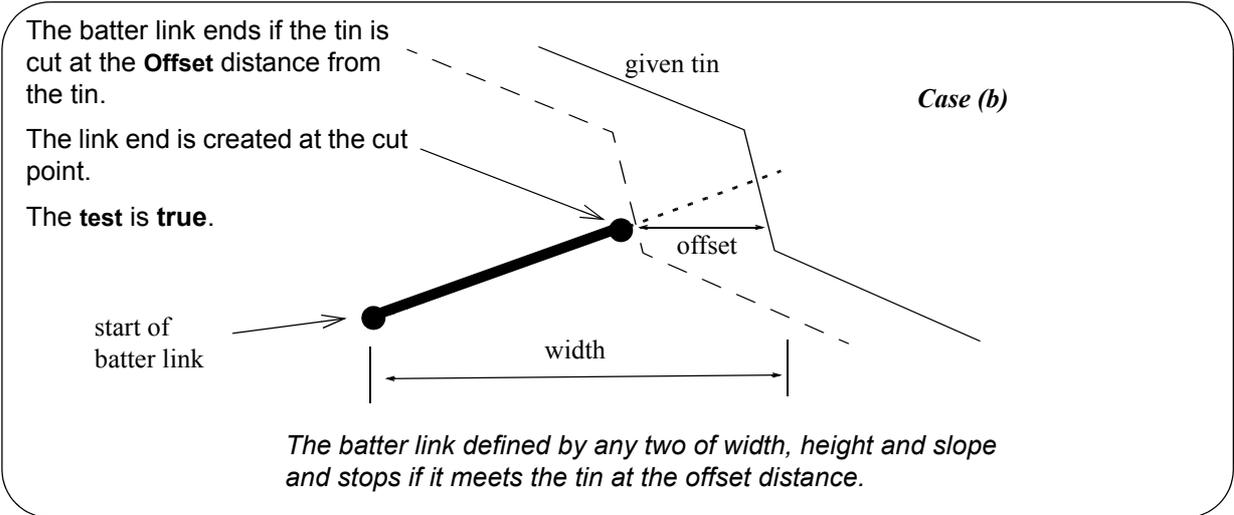
If the link does stop, control is transferred via a **Goto label**, otherwise control continues onto the next line in the table.

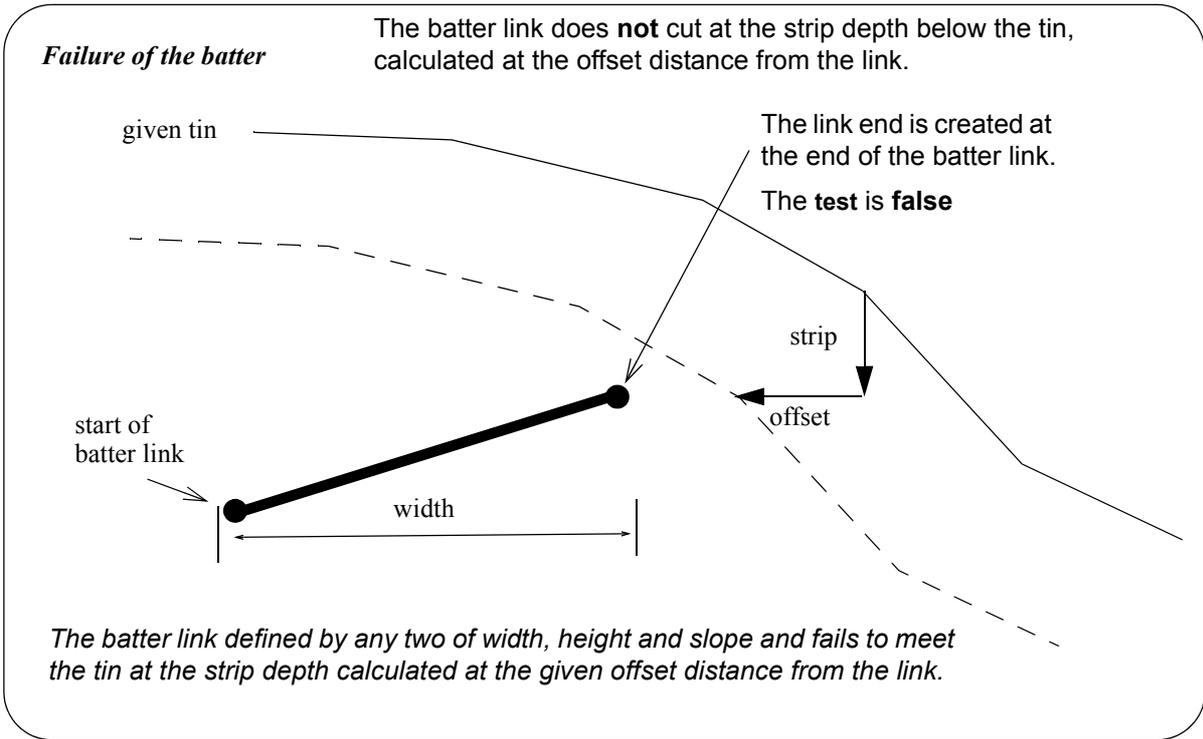
The maximum length of the **batter** link is defined by specifying values for two of the three fields **Width**, **Height** and **Slope**. The **Width**, **Height** and **Slope** values of the created link may be different if the link stops because of one of the conditions (a), (b) or (c).

Notes

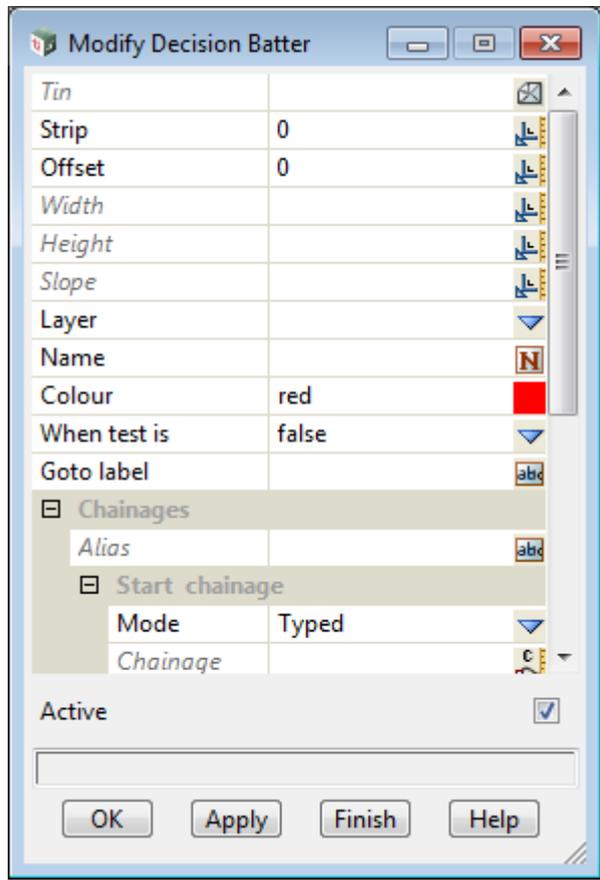
- 1. cases (a) and (b) are just special cases of (c)
- 2. strip and offset can be used to stop the link to allow for a fixed structure (such as a drain) to be inserted so that it ends up on the tin.







Selecting **Batter** brings up the panel **Modify Decision Batter**



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Tin</b> <i>tin to test battering to.</i>	tin box		available tins
<b>Strip</b> <i>distance below the tin to stop at.</i>	real box	0	
<b>Offset</b> <i>offset distance from the link to check strip depth</i>	real box	0	
<b>Width</b> <i>the width for the link. Any two of Width, Height and Slope can be used.</i>	real box		
<b>Height</b> <i>the height for the link. Any two of Width, Height and Slope can be used.</i>	real box		
<b>Slope</b> <i>the slope, in 1v in, of the link. Positive is up and negative down. Any two of Width, Height and Slope can be used.</i>	real box		
<b>Layer</b> <i>name of the layer that the new link is to go in.</i>	layer box	Design	available layers
<b>Name</b> <i>the name to be used for the created link.</i>	name box		available names
<b>Colour</b> <i>the colour to be used for the created link</i>	colour box	cyan	available colours
<b>When test is</b> <i>if <b>When test is</b> has the value <b>true</b>, and the test is <b>true</b> then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>  <i>If <b>When test is</b> has the value <b>false</b>, and the test is <b>false</b> then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>	choice box	false	true, false
<b>Goto label</b> <i>used with <b>When test is</b>.</i> <i>Note that there may be a <b>Then-End Then</b> block that matches the <b>GoTo Label</b>. For a description of a <b>Then-End Then</b> block and how it works, see <a href="#">24.12.4.5.4 Left/Right Modifiers =&gt; Create =&gt;Fixed =&gt;Decisions =&gt;Then</a>.</i>	text box		
<b>Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply</b> <i>the batter is performed when the chainage is between the Start and End chainages.</i> <i>For information on these panel fields, see <a href="#">24.12.2 Common Fields on Modifier Panels</a>.</i>			

### 24.12.4.5.7 Left/Right Modifiers => Create =>Fixed =>Decisions =>Batter test

This option is new in V11.

The **Batter test** option **tests** if a user defined link comes within

- (a) a **Strip** depth of a **Tin**
- (b) an **Offset** distance from a **Tin**.
- (c) a **Strip** depth of a **Tin** calculated at a given **Offset** from the link.

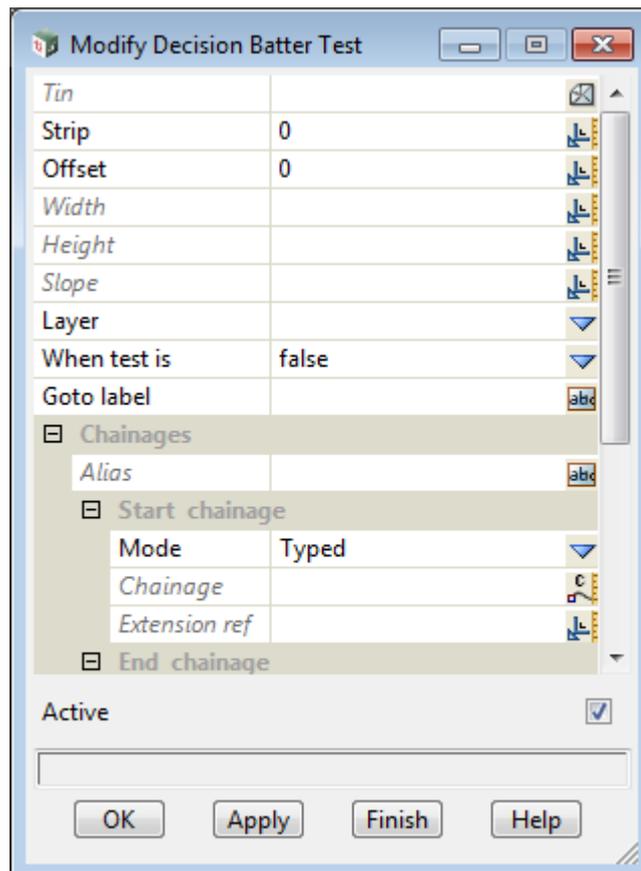
The **test link** is defined by specifying values for two of the three fields **Width**, **Height** and **Slope**.

**Notes**

1. The tests are the same as in [24.12.4.5.6 Left/Right Modifiers => Create =>Fixed =>Decisions =>Batter](#) but **no link is created**. That is, The batter test is used to test if a batter will stop/not stop without creating a batter link.
2. Cases (a) and (b) are just special cases of (c)

For diagrams describing the tests, see [24.12.4.5.6 Left/Right Modifiers => Create =>Fixed =>Decisions =>Batter](#).

Selecting **Batter test** brings up the **Modify Decision Batter Test** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Tin</b>	tin box		available tins
	<i>tin to test battering to.</i>		

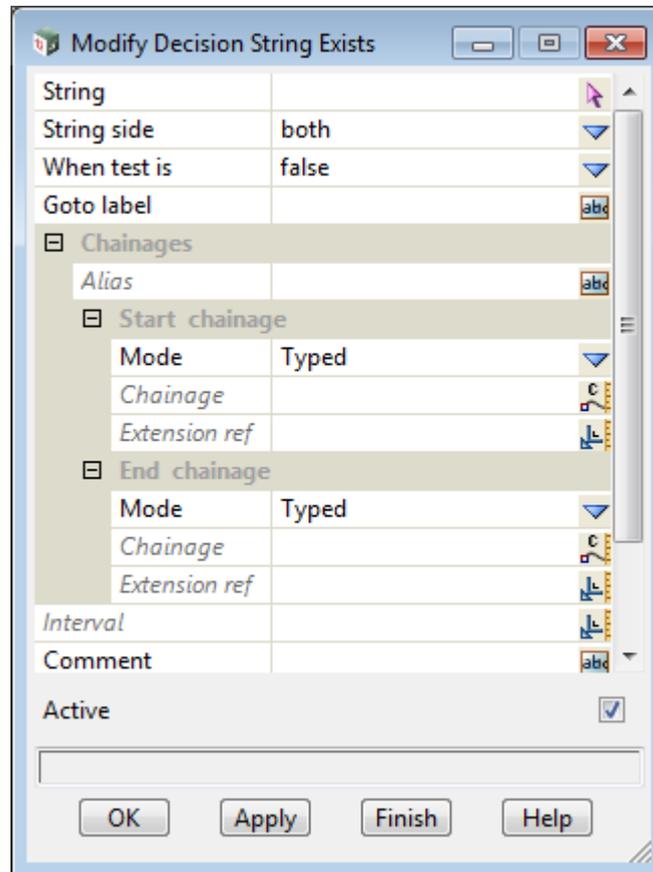
<b>Strip</b>	real box	0	
	<i>distance below the tin to stop at.</i>		
<b>Offset</b>	real box	0	
	<i>offset distance from the link to check strip depth</i>		
<b>Width</b>	real box		
	<i>the width for the link. Any two of Width, Height and Slope can be used.</i>		
<b>Height</b>	real box		
	<i>the height for the link. Any two of Width, Height and Slope can be used.</i>		
<b>Slope</b>	real box		
	<i>the slope, in 1v in, of the link. Positive is up and negative down. Any two of Width, Height and Slope can be used.</i>		
<b>Layer</b>	layer box	Design	available layers
	<i>name of the layer that the test link is appended to. For information on Layers, see <a href="#">24.1.2 MTF Links and Layers</a>.</i>		
<b>When test is</b>	choice box	false	true, false
	<i>if <b>When test is</b> has the value <b>true</b>, and the test is <b>true</b> (i.e. the test link comes within the depth <b>Strip</b> and offset <b>Offset</b> of the tin) then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>		
	<i>If <b>When test is</b> has the value <b>false</b>, and the test is <b>false</b> then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>		
<b>Goto label</b>	text box		
	<i>used with <b>When test is</b>.</i>		
	<i>Note that there may be a <b>Then-End Then</b> block that matches the <b>GoTo Label</b>. For a description of a <b>Then-End Then</b> block and how it works, see <a href="#">24.12.4.5.4 Left/Right Modifiers =&gt; Create =&gt;Fixed =&gt;Decisions =&gt;Then</a>.</i>		
<b>Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply</b>			
	<i>the batter is performed when the chainage is between the Start and End chainages.</i>		
	<i>For information on these panel fields, see <a href="#">24.12.2 Common Fields on Modifier Panels</a>.</i>		

### 24.12.4.5.8 Left/Right Modifiers => Create =>Fixed =>Decisions =>String exists

This option is new in V11.

This options tests to see if a string is on the left and/or right of the reference string.

Selecting **String** brings up the panel **Modify Decision String Exists**



The fields and buttons used in this panel have the following functions.

Field	Description	Type	Default	Pop-Up
<b>String</b>	<i>the string to test.</i>	string select		

<b>String side</b>	<i>if <b>left</b>, only the left side of the reference string is searched for the given string. If <b>right</b>, only the right side of the reference string is searched for the given string. If <b>both</b>, the left and the right side of the reference string is searched for the given string.</i>	choice box	both	left, right, both
--------------------	--	------------	------	-------------------

<b>When test is</b>	<i>if <b>When test is</b> has the value <b>true</b>, and the test is <b>true</b> (i.e. the given string is found) then go to the label <b>Goto label</b>. Otherwise proceed to the next row.  <i>If <b>When test is</b> has the value <b>false</b>, and the test is <b>false</b> (i.e. the given string is not found) then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i></i>	choice box	false	true, false
---------------------	--	------------	-------	-------------

<b>Goto label</b>	<i>used with <b>When test is</b>. Note that there may be a <b>Then-End Then</b> block that matches the <b>GoTo Label</b>. For a description of a <b>Then-End Then</b> block and how it works, see <a href="#">24.12.4.5.4 Left/Right Modifiers =&gt; Create =&gt;Fixed</a></i>	text box		
-------------------	--	----------	--	--

[=>Decisions =>Then.](#)

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*the test is performed when the chainage is between the Start and End chainages.*

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels.](#)*

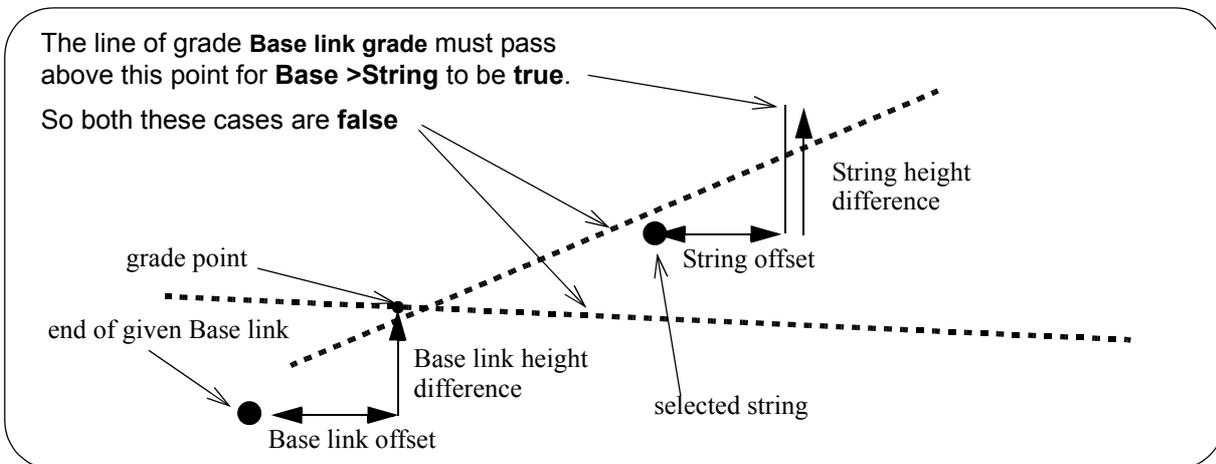
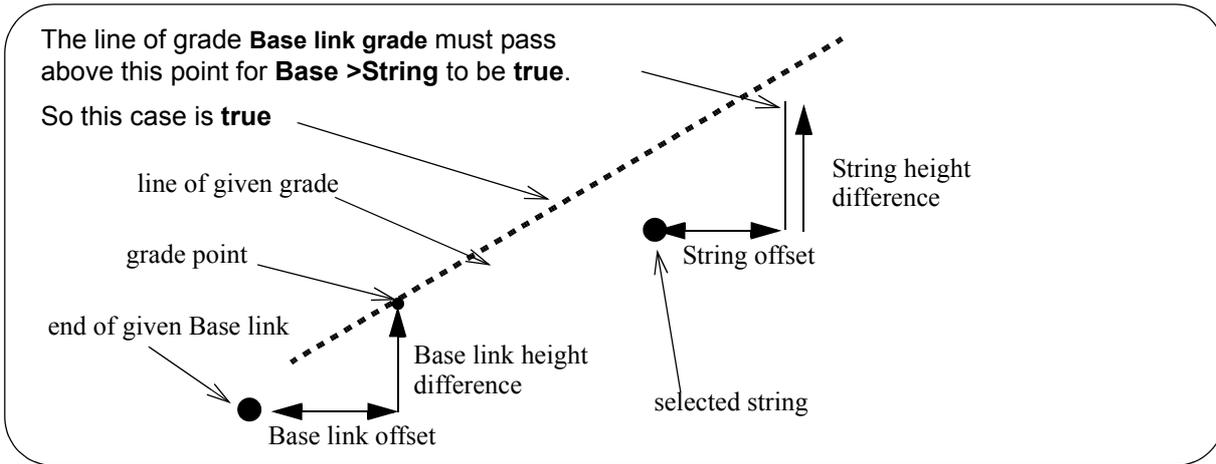
### 24.12.4.5.9 Left/Right Modifiers => Create =>Fixed =>Decisions =>Above String Height

This option is new in V11.

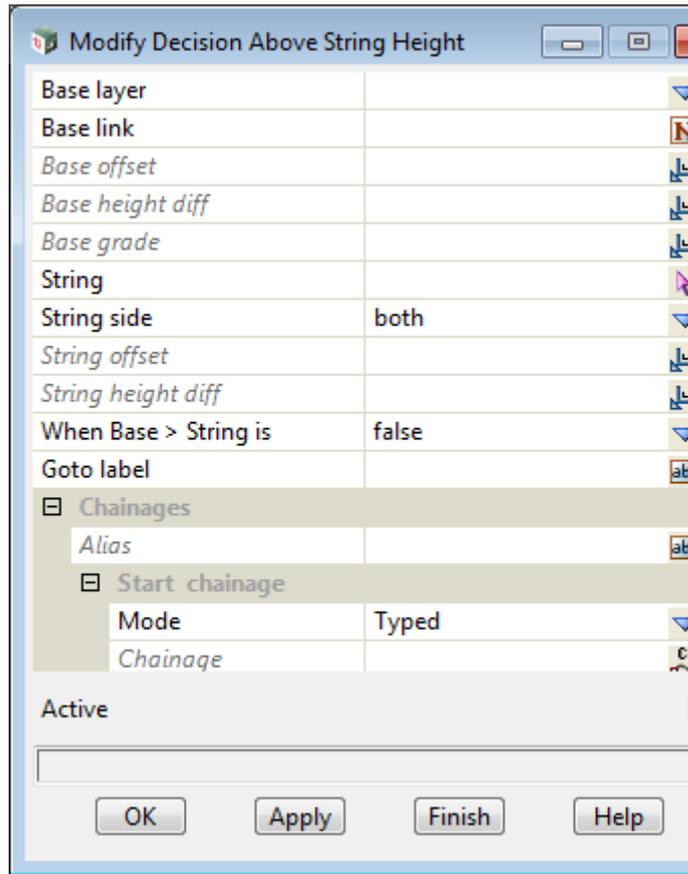
The option **tests** if a line of a given grade (**grade line**) defined from the end of a Base link passes a given height difference above a selected string.

The **grade line** passes through the **grade point** which is a given **Base link offset** and **Base link height difference** from the end of the given link.

The height is tested at a given **string offset** from the selected string.



Selecting **Above string height** brings up the **Modify Decision Above String Height** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Base layer</b> <i>name of the layer that the link to test for is in. For information on Layers.</i>	layer box	Design	available layers
<b>Base link</b> <i>name of the link in <b>Base layer</b> to test if the line of grade going from the link is a delta height above a selected string.</i>	link box		
<b>Base link offset</b> <i>option "sided" offset distance from the end of the link <b>Base link</b> that the line of grade <b>Base link grade</b> goes through (the grade point).</i>	real box		
<b>Base link height diff</b> <i>the value to add to the height of the end of the link <b>Base link</b> that the line of grade <b>Base link grade</b> goes through (the grade point).</i>	real box		
<b>Base link grade</b> <i>percentage grade of a line through the grade point that is tested to see if it passed withing a given offset and height difference of the selected string.</i>	real box		
<b>String</b> <i>the string to test.</i>	string select		
<b>String side</b> <i>if <b>left</b>, only the left side of the reference string is searched for the selected string. If <b>right</b>, only the right side of the reference string is searched for the selected string. If <b>both</b>, the left and the right side of the reference string is searched for the selected string.</i>	choice box	both	left, right, both

**String offset**                      real box

*optional "absolute" offset distance from the string of the point that is tested for the **grade link** going above.*

**String height diff**                real box

*optional height difference added to the height of the string. This total height is used at the String offset point to give the point that is tested for the **grade link** going above.*

**When Base >String is**            choice box                      false                      true, false

*if **When Base >String** has the value **true**, and the test is **true** then go to the label **Goto label**. Otherwise proceed to the next row.*

*If **When Base > String** has the value **false**, and the test is **false** then go to the label **Goto label**. Otherwise proceed to the next row.*

**Goto label**                            text box

*used with **When Base > String is**.*

*Note that there may be a **Then-End Then** block that matches the **GoTo Label**. For a description of a **Then-End Then** block and how it works, see [24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then](#).*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*the test is performed when the chainage is between the Start and End chainages.*

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

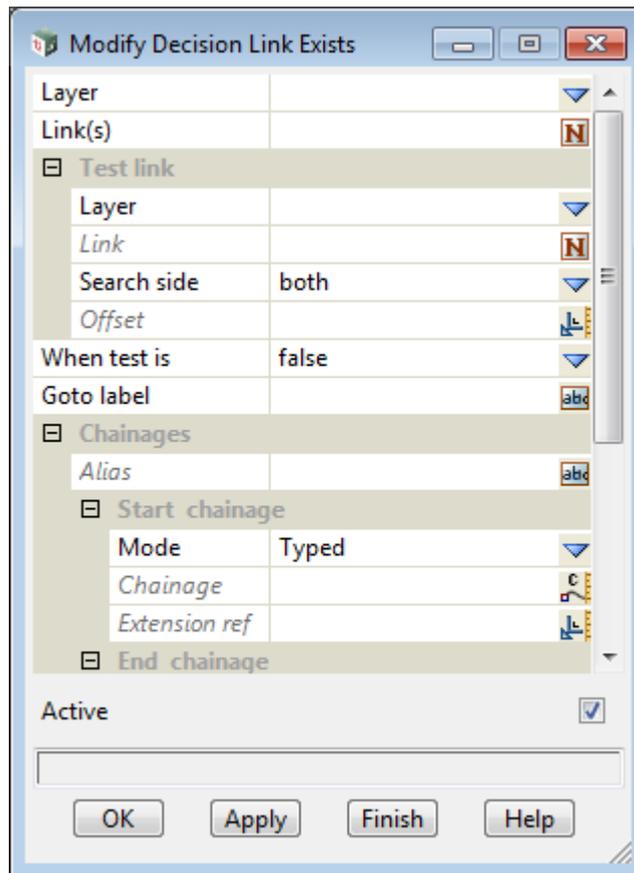
### 24.12.4.5.10 Left/Right Modifiers => Create =>Fixed =>Decisions =>Link Exists

This option is new in V11.

This options tests to see if a given link exists.

**Note** that a link is only looked for on the side that the Link Exists command being processed is in.

Selecting **Link** brings up the panel **Modify Decision Link Exists**



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Layer</b>	layer box	Design	available layers
<b>Link</b>	input		
<i>name of the link in <b>Layer</b> to test if it exists. The test only looks at the side that the Link Exists command being processed is in. processed (Left or Right side).</i>			
<b>When test is</b>	choice box	false	true, false
<i>if <b>When test is</b> has the value <b>true</b>, and the test is <b>true</b> (i.e. the given link is found) then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>			
<i>If <b>When test is</b> has the value <b>false</b>, and the test is <b>false</b> (i.e. the given link is <b>not</b> found) then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>			

**Goto label**                      text box

*used with **When test is**.*

*Note that there may be a **Then-End Then** block that matches the **GoTo Label**. For a description of a **Then-End Then** block and how it works, see [24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then](#).*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*the test is performed when the chainage is between the Start and End chainages.*

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

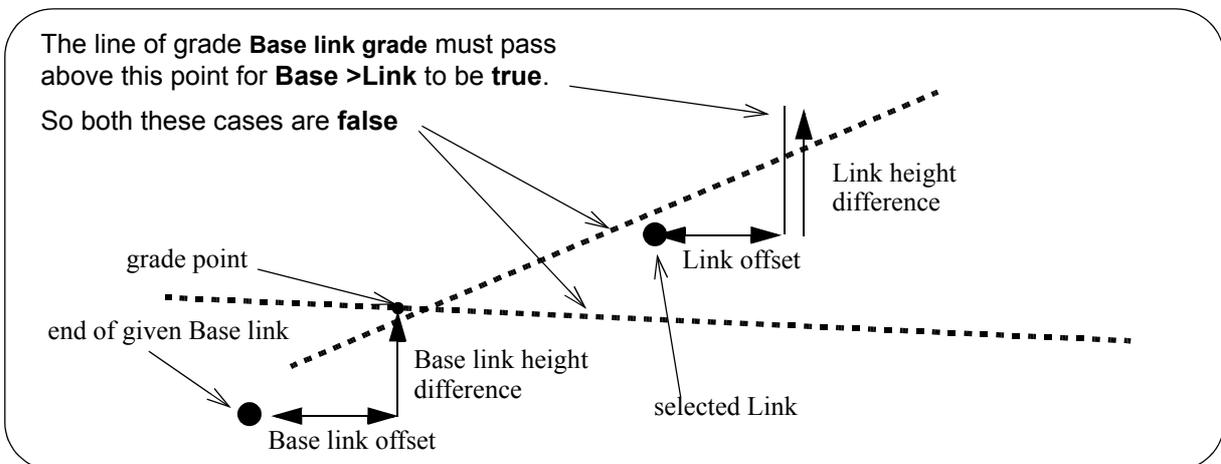
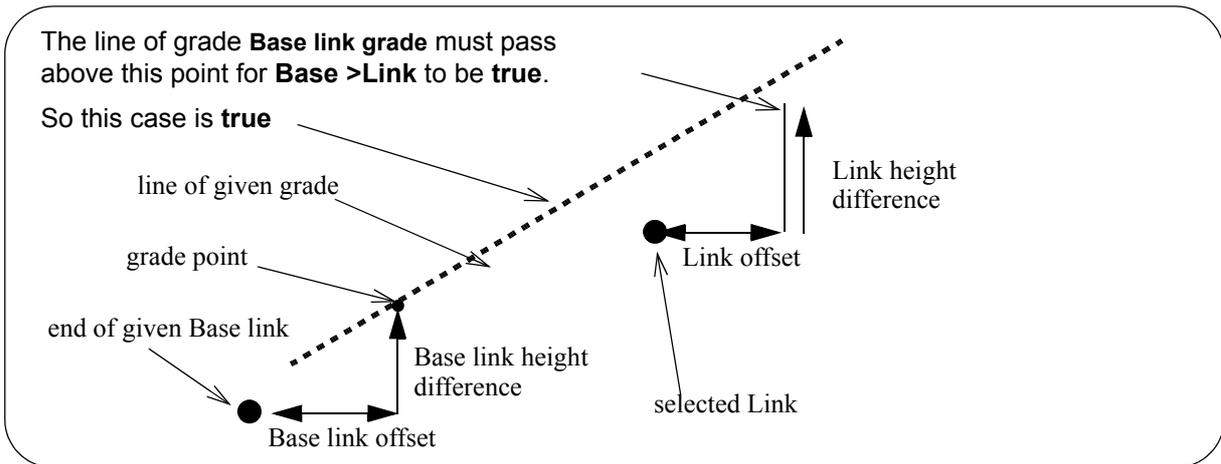
### 24.12.4.5.11 Left/Right Modifiers => Create =>Fixed =>Decisions =>Above Link Height

This option is new in V11.

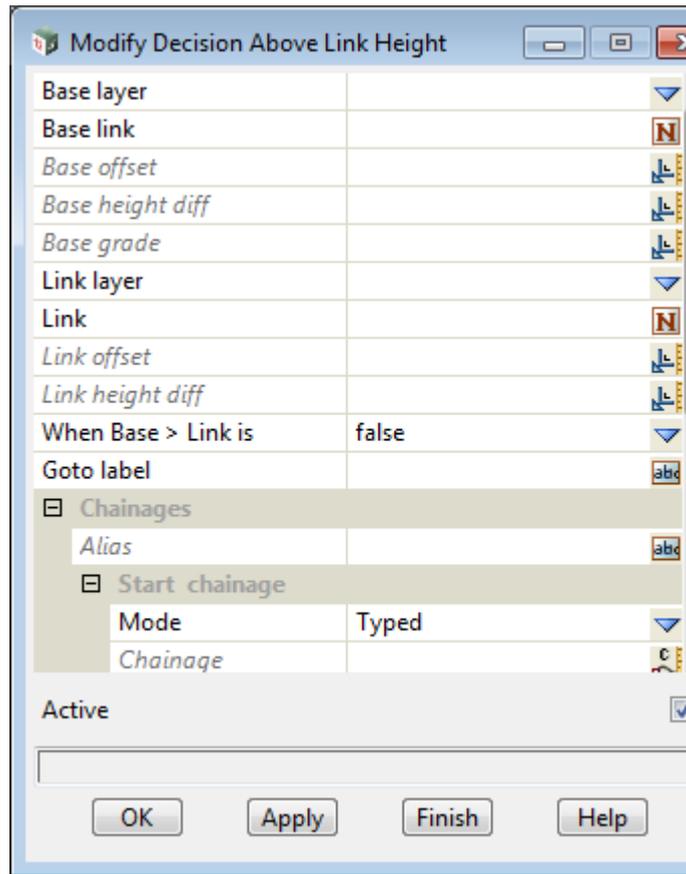
The option **tests** if a line of a given grade (**grade line**) defined from the end of a **Base link** passes a given height difference above a selected **Link**.

The **grade line** passes through the **grade point** which is a given **Base link offset** and **Base link height difference** from the end of the given link.

The height is tested at a given **Link offset** from the selected **Link**.



Selecting **Above link height** brings up the **Modify Decision Above Link Height** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Base layer</b> <i>name of the layer that the base link to test for is in. For information on Layers.</i>	layer box	Design	available layers
<b>Base link</b> <i>name of the base link in <b>Base layer</b> to test if the line of grade going from the base link is a delta height above a selected <b>Link</b>.</i>	link box		
<b>Base link offset</b> <i>option "sided" offset distance from the end of the link <b>Base link</b> that the line of grade <b>Base link grade</b> goes through (the grade point).</i>	real box		
<b>Base link height diff</b> <i>the value to add to the height of the end of the link <b>Base link</b> that the line of grade <b>Base link grade</b> goes through (the grade point).</i>	real box		
<b>Base link grade</b> <i>percentage grade of a line through the grade point that is tested to see if it passed above a given offset and height difference of a selected <b>Link</b>.</i>	real box		
<b>Link layer</b> <i>name of the layer that the link to test that the grade line is above, is in. For information on Layers.</i>	layer box	Design	available layers
<b>Link</b> <i>name of the link in <b>Layer</b> to test if the line of grade going from the base link is a delta height above it.</i>	link box		
<b>Link offset</b>	real box		

optional "absolute" offset distance from the end of **Link** of the point that is tested for the **grade link** going above.

**Link height diff**                      real box

optional height difference added to the height of the end of **Link**. This total height is used at the **Link** offset point to give the point that is tested for the **grade link** going above.

**When Base >String is**              choice box              false                      true, false

if **When Base >Link** has the value **true**, and the test is **true** then go to the label **Goto label**. Otherwise proceed to the next row.

If **When Base > Link** has the value **false**, and the test is **false** then go to the label **Goto label**. Otherwise proceed to the next row.

**Goto label**                              text box

used with **When Base > Link is**.

Note that there may be a **Then-End Then** block that matches the **GoTo Label**. For a description of a **Then-End Then** block and how it works, see [24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then](#).

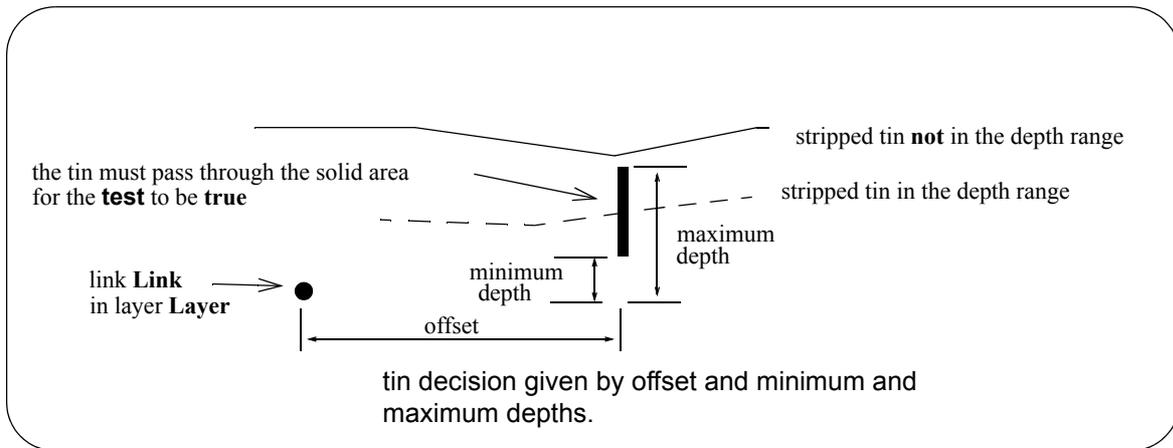
### 24.12.4.5.12 Left/Right Modifiers => Create =>Fixed =>Decisions =>Tin

This option is new in V11.

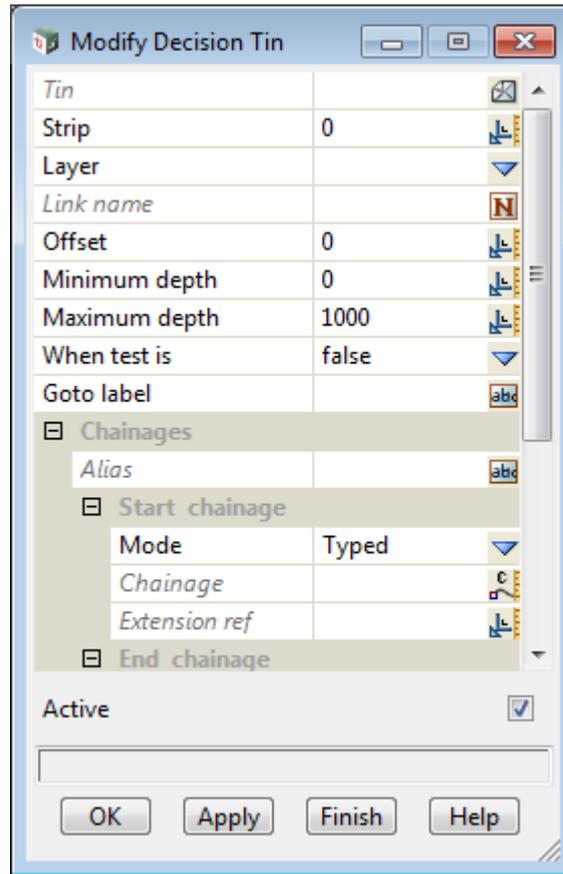
Selecting **Tin** brings up the **Modify Decision Tin** panel which tests to see if the depth from the end of the given **Link** in a given **Layer**, offset by the amount given in the **Offset** field, is between the two values given in the **Minimum depth** and **Maximum depth** fields where the depths are taken from the stripped Tin (that is, the **Tin** dropped by the **Strip** value).

If the depth is between the values, then control is transferred to the row of the **Modifiers** grid with the label given by the **Goto label** field. Otherwise, control passes to the next row of the **Modifiers** grid.

**Offset**, **Minimum depth** and **Maximum depth** can be positive or negative.



Selecting **Tin** brings up the panel **Modify Decision Tin**



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Tin</b> <i>the tin to calculate the depth to.</i>	tin box		available tins
<b>Strip</b> <i>the depth to drop the tin before testing.</i>	real box	0	
<b>Layer</b> <i>name of the layer that the link to test is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>name of link in <b>Layer</b> to test against the tin.</i>	input		
<b>Offset</b> <i>the depth is calculate at an offset distance of <b>offset</b> from the end of the previous link.</i>	real box	0	
<b>Minimum depth</b> <i>if the depth is between the minimum and maximum depth, then control is passed to the line with the label given in the goto field, otherwise control passes onto the next line of the table.</i>	real box	0	
<b>Maximum depth</b> <i>see previous field.</i>	real box	1000	
<b>When test is</b> <i>if <b>When test is</b> has the value <b>true</b>, and the test is <b>true</b> then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>  <i>If <b>When test is</b> has the value <b>false</b>, and the test is <b>false</b> (i.e. the given link is <b>not</b> found) then go to the</i>	choice box	false	true, false

label **Goto label**. Otherwise proceed to the next row.

**Goto label**                      text box

used with **When test is**.

Note that there may be a **Then-End Then** block that matches the **GoTo Label**. For a description of a **Then-End Then** block and how it works, see [24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then](#).

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

the test is performed when the chainage is between the Start and End chainages.

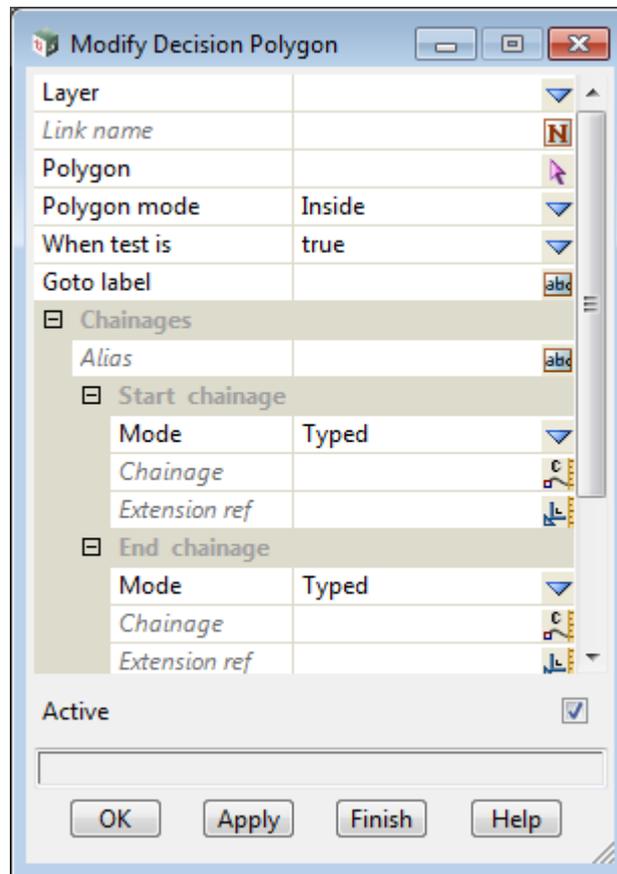
For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

### 24.12.4.5.13 Left/Right Modifiers => Create =>Fixed =>Decisions =>Polygon

This option is new in V11.

This options tests to see if the (x,y) coordinates of a link is inside a user selected polygon.

Selecting **Polygon** brings up the panel **Modify Decision Polygon**



The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>Layer</b> <i>name of the layer that the link to test is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>name o f link in <b>Layer</b> to test against the polygon.</i>	input		
<b>Polygon</b> <i>the polygon to use for testing if the link is inside or outside of it.</i>	string sect		
<b>Polygon mode</b> <i>if <b>Inside</b>, the test is true if the (x,y) coordinates of the link is inside the polygon. If <b>Outside</b>, the test is true if the (x,y) coordinates of the link is outside the polygon</i>	choice box	Inside	Inside, Outside
<b>When test is</b> <i>if <b>When test is</b> has the value <b>true</b>, and the test is <b>true</b> then go to the label <b>Goto label</b>. Otherwise proceed to the next row.  If <b>When test is</b> has the value <b>false</b>, and the test is <b>false</b> (i.e. the given link is <b>not</b> found) then go to the label <b>Goto label</b>. Otherwise proceed to the next row.</i>	choice box	false	true, false

**Goto label**                      text box

*used with **When test is**.*

*Note that there may be a **Then-End Then** block that matches the **GoTo Label**. For a description of a **Then-End Then** block and how it works, see [24.12.4.5.4 Left/Right Modifiers => Create =>Fixed =>Decisions =>Then](#).*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*the test is performed when the chainage is between the Start and End chainages.*

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.6 Left/Right Modifiers => Create =>Fixed =>from link

The **Fixed =>from link** option can take the width, height and xfall from another link.

It doesn't matter which of the two values from width, height xfall or slope were used to defined the selected link, the **from link** option take one value **from** the specified link whilst holding another one to be what it is defined as by the current link.



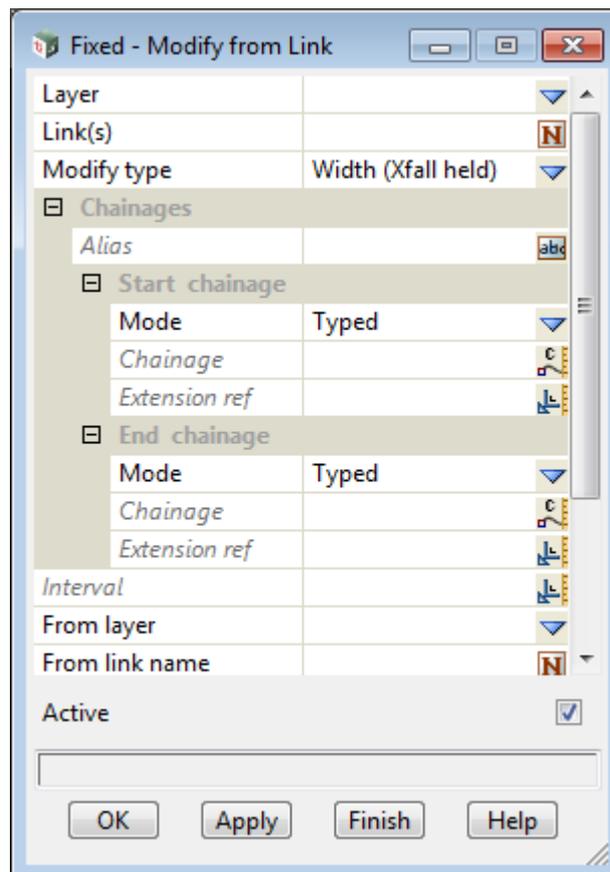
So the **from link** option replaces the V10 **Fixed** options

**Fixed =>from link =>Width from link**

**Fixed =>from link =>Height from link**

**Fixed =>from link =>Xfall from link**

Selecting the **From link** brings up the **Fixed - Modify from Link** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Layer** layer box Design available layers  
*name of the layer that the link to be modified is in.*

**Link name** name box names.4d pop-up  
*the name of the link in the Layer to modify the width/height/xfall/slope using the method given in **Modifier type**, between the chainages given by **Start mode** and **End mode**.*  
*This is the current link.*

**Modifier type** choice box

modify Width to be as from selected link, take Height from current link  
 modify Width to be as from selected link, take Xfall from current link  
 modify Height to be as from selected link, take Width from current link  
 modify Height to be as from selected link, take Xfall from current link  
 modify Xfall to be as from selected link, take With from current link  
 modify Xfall to be as from selected link, take Height from current link



*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).*

*The **Modify** width/height/xfall is the part of the selected link that is modified from its current value by the method given by **Modifier type**.*

*The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.*

*For **Modifier type Modify Width, Hold Height/Xfall**, the width is taken from the link **From link name** that is in the layer **From layer** and zone **From zone**, and the **Height/Xfall** is taken from the current link.*

*For **Modifier type Modify Height, Hold Width/Xfall**, the height is taken from the link **From link name** that is in the layer **From layer** and zone **From zone**, and the **Width/Xfall** is taken from the current link.*

*For **Modifier type Modify Xfall, Hold Width/Height**, the xfall is taken from the link **From link name** that is in the layer **From layer** and zone **From zone**, and the **Width/Height** is taken from the current link.*

**From layer** layer box Design available layers  
*name of the layer that the link to take values from is in.*

**From Link name** name box names.4d pop-up  
*the name of the link in the **From Layer** and **From Zone** to get the width/height/xfall from.*

**From zone** choice box fixed, cut, fill  
*zone that the **From link name** link is in.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**  
*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.7 Left/Right Modifiers => Create =>Fixed =>to string

The **Fixed =>to string** option can calculate the width, height and xfall of a link by going from the start point of the link to another selected string.

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and xfall were used to defined the fixed link, the **to string** option allows for any combination to specify how to **modify** one value whilst holding another one to what it is defined as by the link.



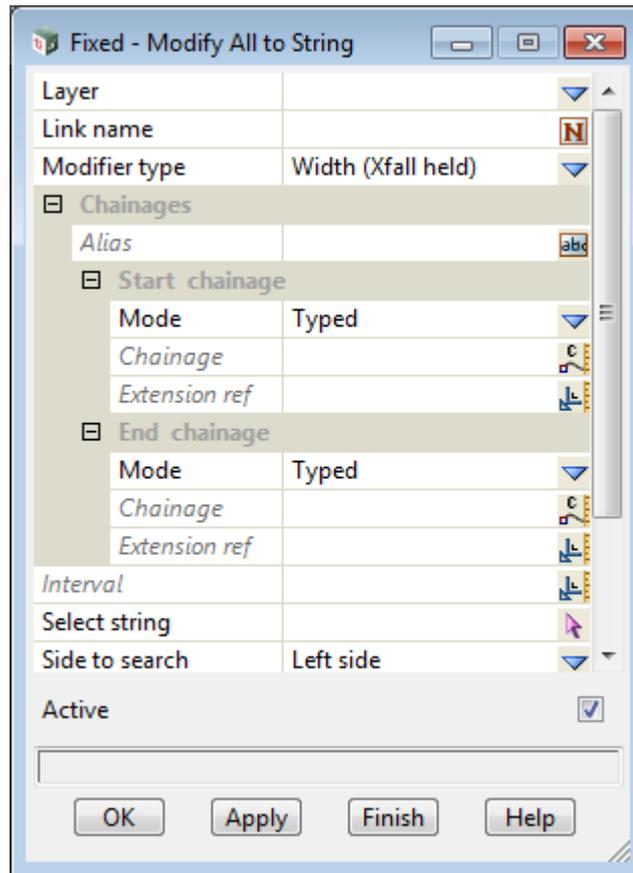
So the **to string** option replaces the V10 **Fixed** options

**Fixed =>to string =>Width to string**

**Fixed =>to string =>Height to string**

**Fixed =>to string =>Xfall to string**

Selecting the **To** string brings up the **Fixed - Modify All to String** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. This is the current link.</i>	name box		names.4d pop-up
<b>Modifier type</b>  modify Width and take Height from selected link modify Width and take Xfall from selected link modify Height and take Width from selected link modify Height and take Xfall from selected link modify Xfall and take With from selected link modify Xfall and take Height from selected link modify Width, Height and Xfall to get onto string	choice box		



A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).

The **Modify** width/height/xfall is the part of the selected link that is modified from its current value by

the method given by **Modifier type**.

The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.

For **Modify Width**, see [24.12.4.7.1 Fixed =>to String Modifier Types "Modify Width. Hold Height/Xfall"](#)

For **Modify Height**, see [24.12.4.7.2 Fixed =>to String Modifier Types "Modify Height. Hold Width/Xfall"](#)

For **Modify Xfall**, see [24.12.4.7.3 Fixed => to String Modifier Types "Modify Xfall. Hold Width/Height"](#)

For **Modify Width, Height and Xfall**, see [24.12.4.7.4 Fixed => to String Modifier Type "Modify Width, Height and Xfall"](#)

**Select string**                      string-select

*select string to use for defining width/height/xfall for the link.*

**Side to search**                      choice box                      left side                      left side, right side, both sides

*side of the hinge string to start searching to find the string to define width/height/xfall.*

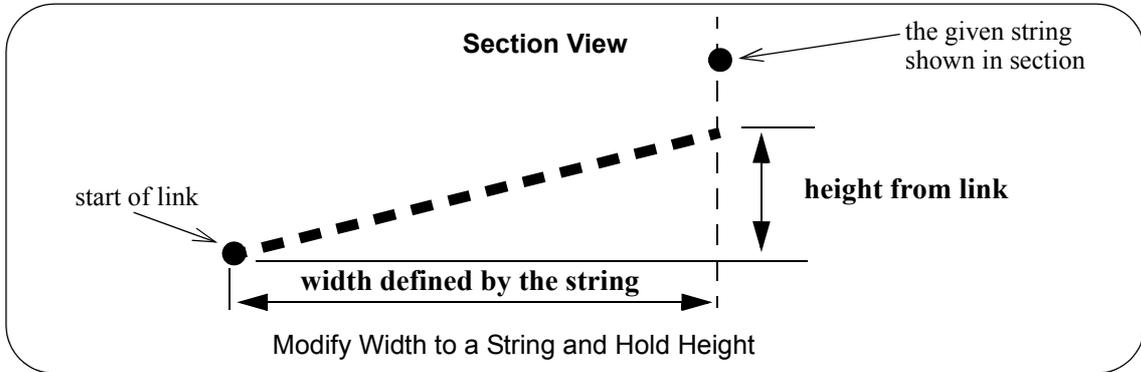
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .*

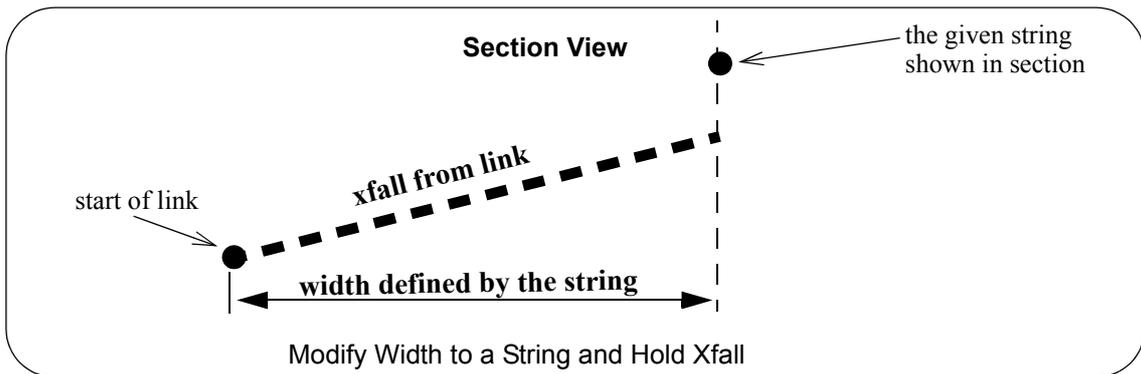
### 24.12.4.7.1 Fixed =>to String Modifier Types "Modify Width, Hold Height/Xfall"

For a fixed link **Modify Width** calculates the **width** of the link as the width from the start point of the link, **to the selected string**.

For **Modify Width, Hold Height** the *height* is taken from the link.



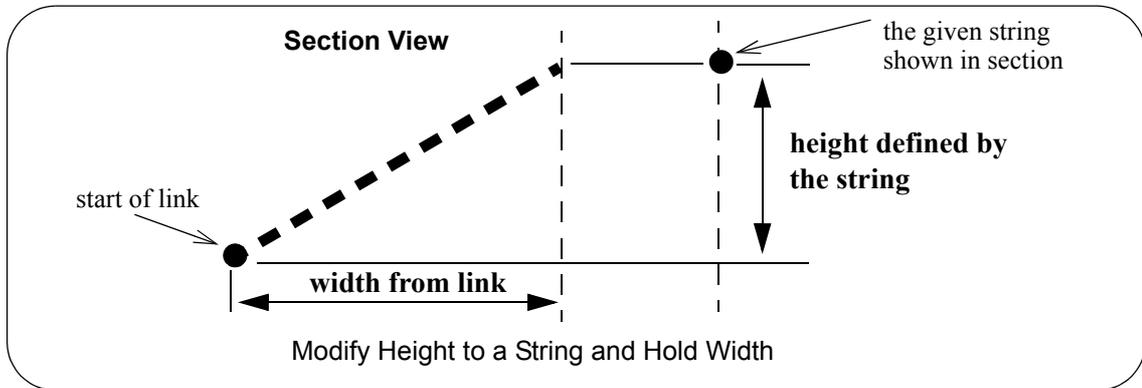
For **Modify Width, Hold Xfall** the *xfall* is taken from the link.



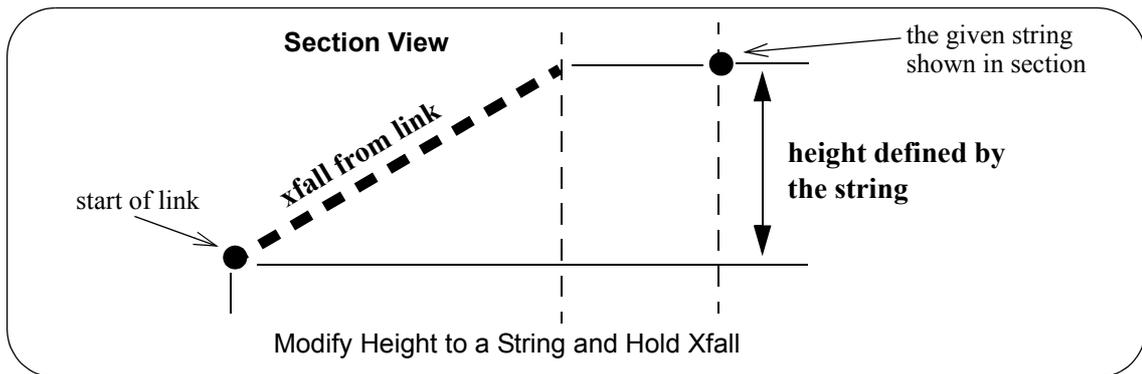
### 24.12.4.7.2 Fixed =>to String Modifier Types "Modify Height, Hold Width/Xfall"

For a fixed link **Modify Height** calculates the **height** of the link as the difference in the height at the start point of the link, and the height **at the selected string**.

For **Modify Height, Hold Width** the *width* is taken from the link.



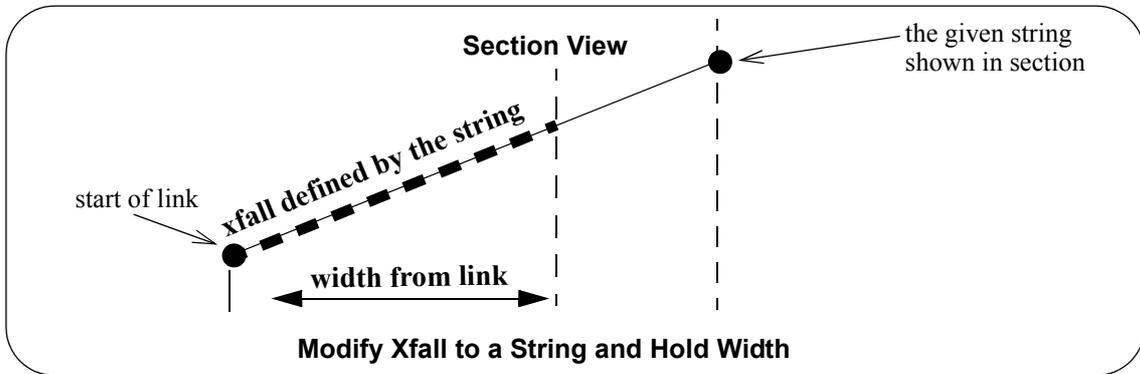
For **Modify Height, Hold Xfall** the *xfall* is taken from the link.



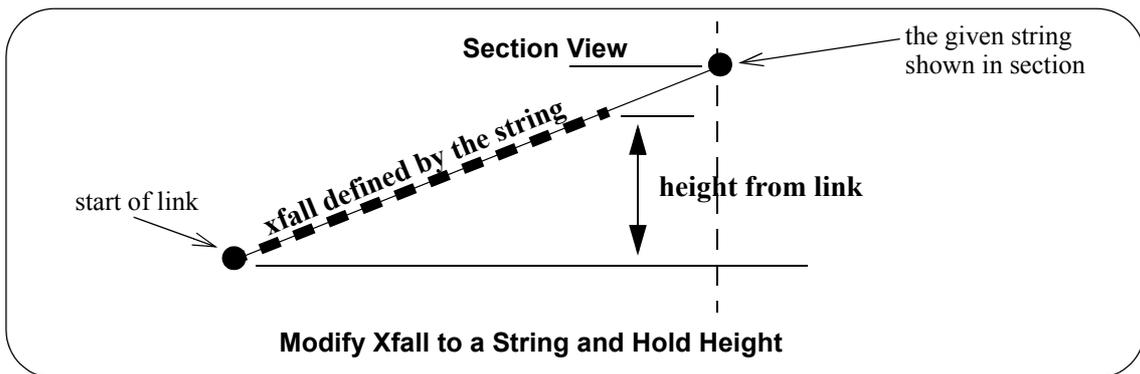
### 24.12.4.7.3 Fixed => to String Modifier Types "Modify Xfall, Hold Width/Height"

For a fixed link **Modify Xfall** calculates the **xfall** of the link as the *xfall* from the start point of the link to the **selected string**.

For **Modify Xfall, Hold Width** the *width* is taken from the link.



For **Modify Xfall, Hold Height** the *height* is taken from the link.



**Note:**

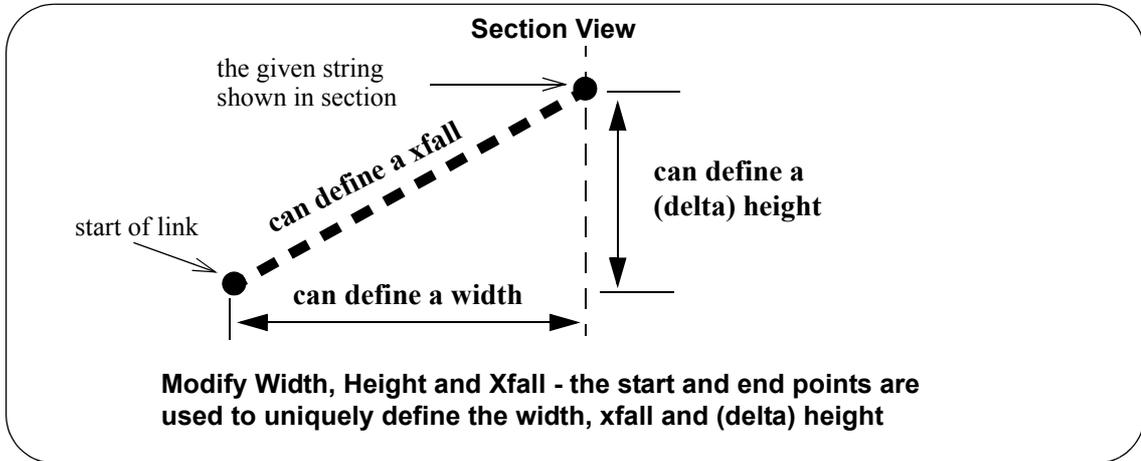
Using two of the above modifiers together with the same string will place the end point of the link on the selected string. For example using the same selected string, **Modify Width, Hold Xfall** followed by a **Modify Xfall, Hold Width** will place the end of the link on that string.

But the **To string** option with the **Modifier type Modify Xfall, Height and Xfall** will do the same thing in one command. See [24.12.4.7.4 Fixed => to String Modifier Type "Modify Width, Height and Xfall"](#).

However, to use the options above you must **first have a link defined** but in V11 there is a new command **Fixed =>Insert =>Insert at string** which creates a link and also places it on a selected string. See [24.12.4.1.3 Left/Right Modifiers => Create =>Fixed =>Insert =>Insert at string](#).

### 24.12.4.7.4 Fixed => to String Modifier Type "Modify Width, Height and Xfall"

For a fixed link **Modify Width, Height and Xfall** calculates the required width, height and/or xfall of the link needed to get from the start point of the link **to the selected string**.



### 24.12.4.8 Left/Right Modifiers => Create =>Fixed =>to tin

The **Fixed =>to tin** option calculates the width, height and xfall or slope of a link so that it is the given number intersection with a given tin. The tin does not have to be the same tin as used in the **Apply MTF** function that is using the MTF.

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and xfall were used to defined the fixed link, the **to tin** option allows for any combination to specify how to **modify** one value to get to the tin whilst holding another one to what it is defined as by the link. In V11 the user is also able to specify the intersection number with the tin to use rather than just taking the first as you can only do in V10.



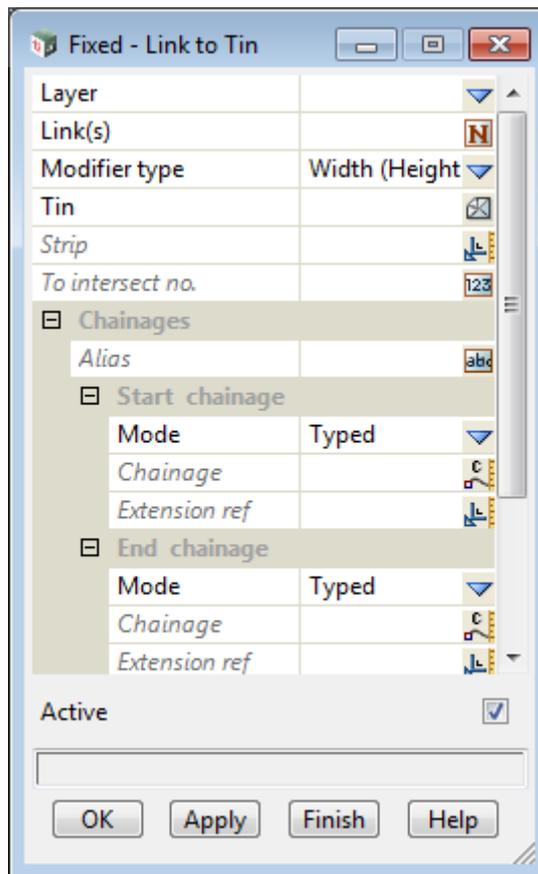
So the **to tin** option replaces the V10 **Fixed** options

**Fixed =>to tin =>Width to tin**

**Fixed =>to tin =>Height to tin**

**Fixed =>to tin =>Xfall to tin**

Selecting the **To tin** brings up the **Fixed - Link to Tin** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. This is the current link.</i>	name box		names.4d pop-up

Modifier type	choice box
modify Width and take Height from current link	
modify Width and take Xfall from current link	
modify Height and take Width from current link	
modify Height and take Xfall from current link	
modify Xfall and take With from current link	
modify Xfall and take Height from current link	
modify Slope and take With from current link	
modify Slope and take Height from current link	



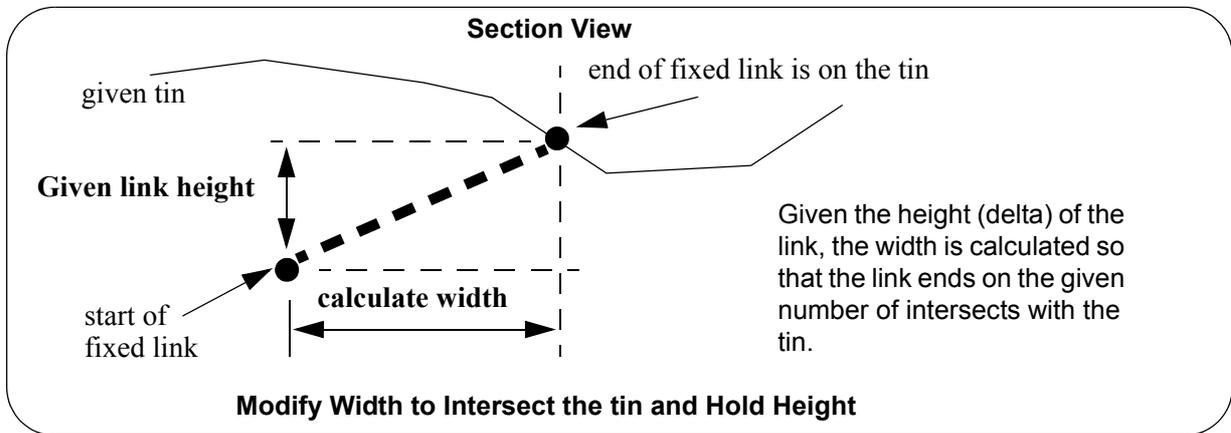
*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#)*



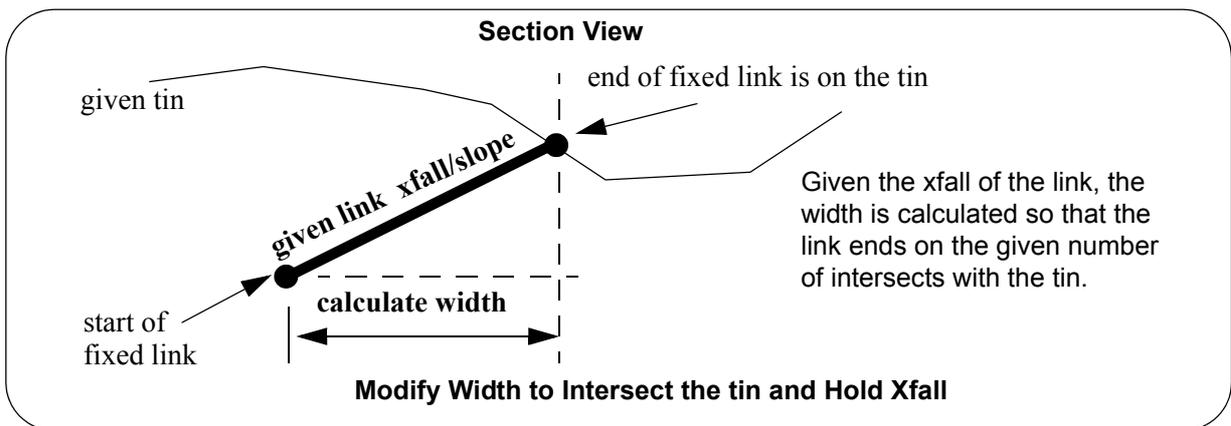
### 24.12.4.8.1 Fixed =>to Tin Modifier Types "Modify Width, Hold Height/Xfall"

For a fixed link **Modify Width** calculates the **width** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Width, Hold Height** the *height* is taken from the link.



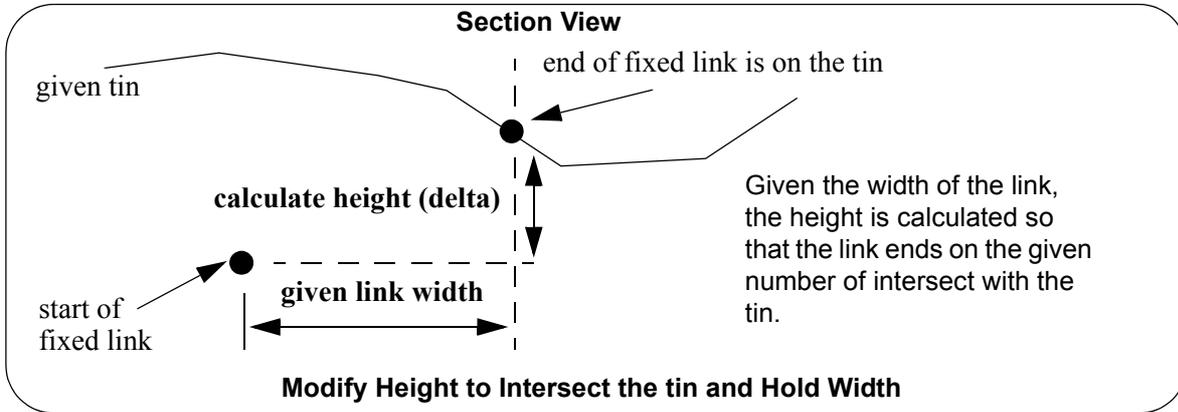
For **Modify Width, Hold Xfall** the *xfall* is taken from the link.



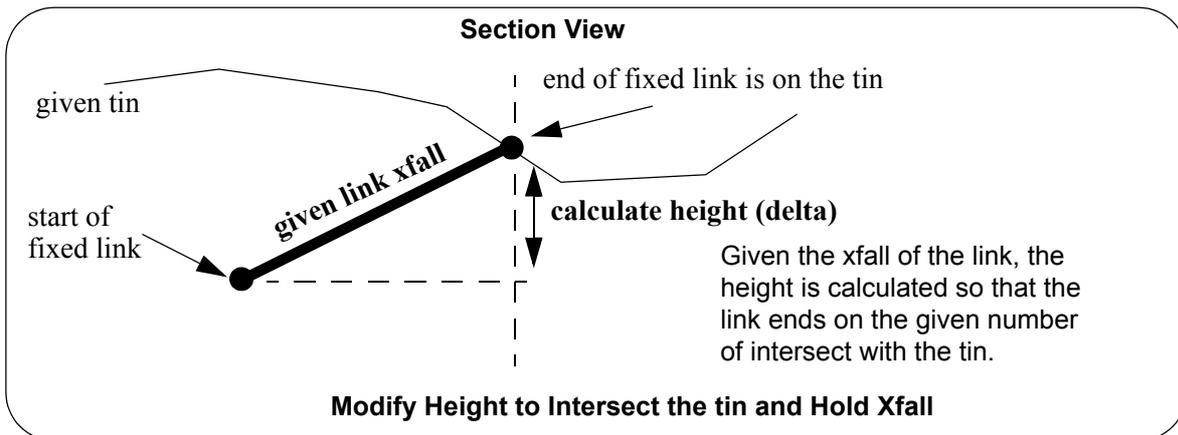
### 24.12.4.8.2 Fixed =>to Tin Modifier Types "Modify Height, Hold Width/Xfall"

For a fixed link **Modify Height** calculates the **height** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Height, Hold Width** the *width* is taken from the link.



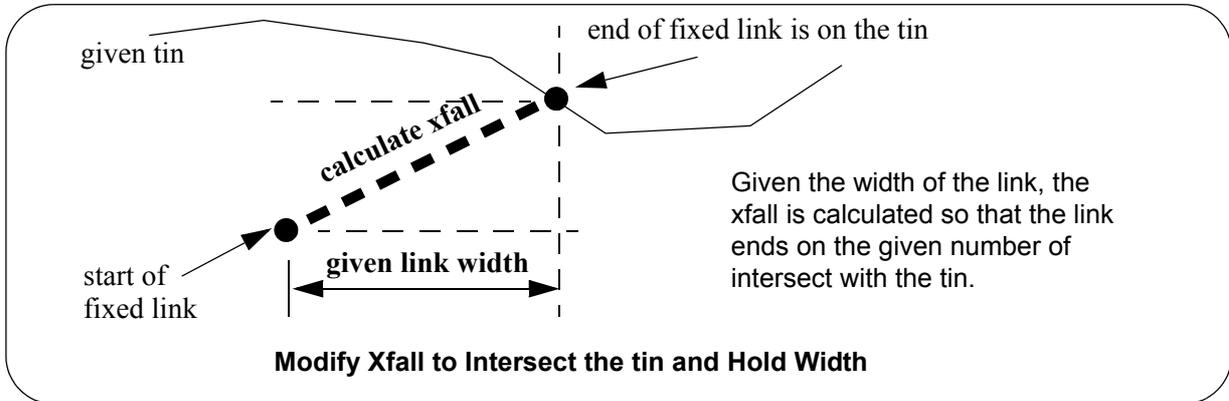
For **Modify Height, Hold Xfall** the *xfall* is taken from the link.



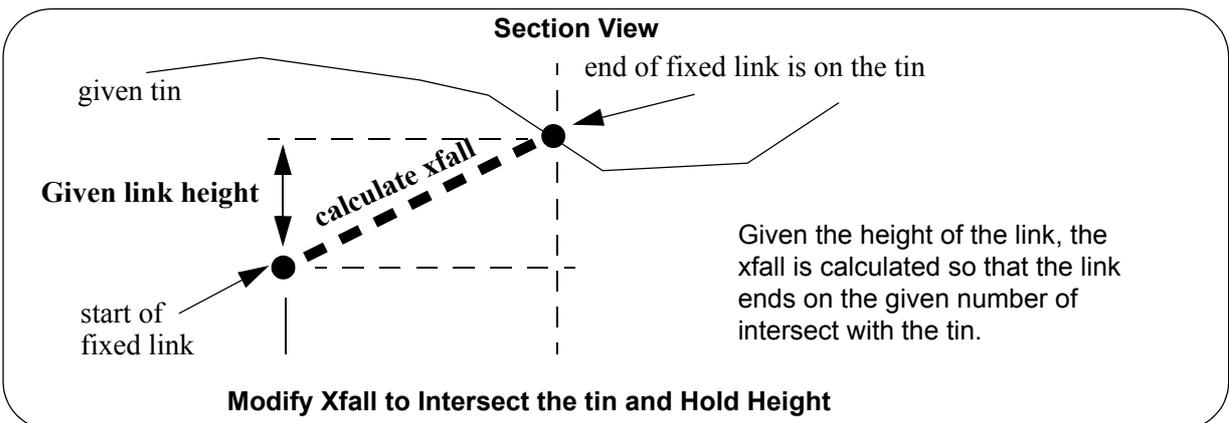
### 24.12.4.8.3 Fixed => to Tin Modifier Types "Modify Xfall, Hold Width/Height"

For a fixed link **Modify Xfall** calculates the **xfall** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Xfall, Hold Width** the *width* is taken from the link.



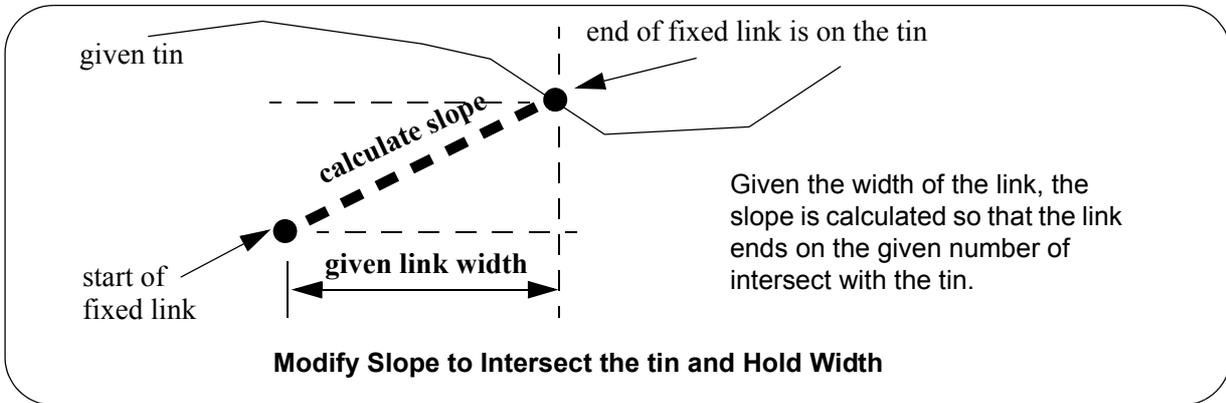
For **Modify Xfall, Hold Height** the *height* is taken from the link.



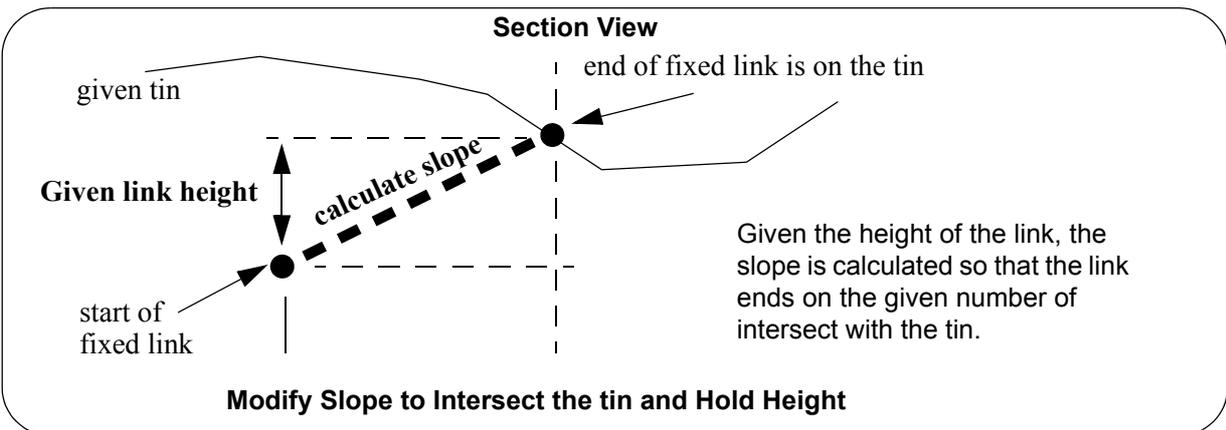
### 24.12.4.8.4 Fixed => to Tin Modifier Types "Modify Slope, Hold Width/Height"

For a fixed link **Modify Slope** calculates the **slope** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Slope, Hold Width** the *width* is taken from the link.



For **Modify Slope, Hold Height** the *height* is taken from the link.



### 24.12.4.9 Left/Right Modifiers => Create =>Fixed =>to RL

The **Fixed =>to RL** option can calculate the width, height and xfall of a link by going from the start point of the link to a given RL (elevation).

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and xfall were used to defined the fixed link, the **to RL** option allows for any combination to specify how to **modify** one value whilst holding another one to what it is defined as by the link.

One restriction is of course that it makes no sense to **Hold Height**.



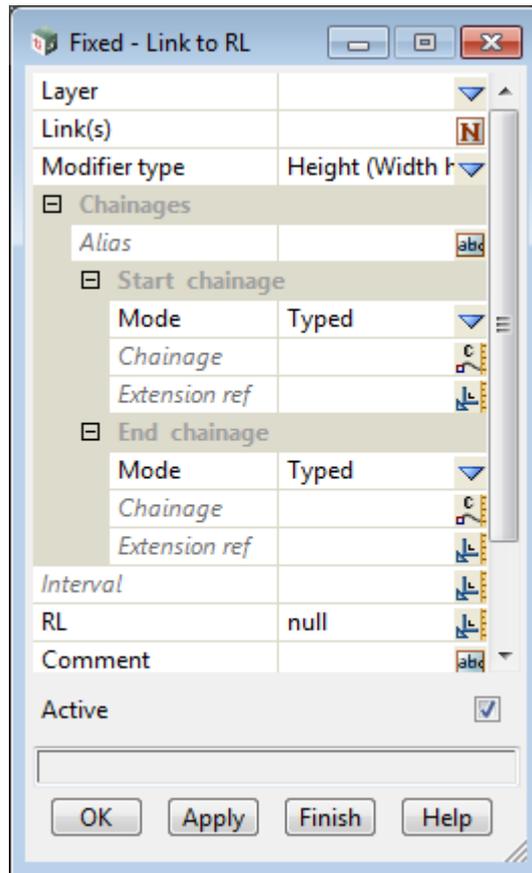
So the **to RL** option replaces the V10 **Fixed** options

**Fixed =>to RL =>Width to RL**

**Fixed =>to RL =>Height to RL**

**Fixed =>to RL =>Xfall to RL**

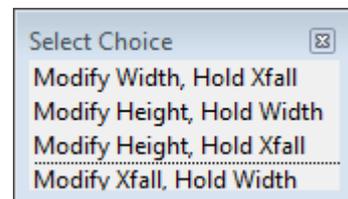
Selecting the **To RL** brings up the **Fixed - Link to RL** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. This is the current link.</i>	name box		names.4d pop-up
<b>Modifier type</b>	choice box		

modify Width and take Xfall from current link  
 modify Height and take Width from current link  
 modify Height and take Xfall from current link  
 modify Xfall and take With from current link



A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#)

The **Modify** width/height/xfall is the part of the selected link that is modified from its current value by the method given by **Modifier type**.

The **Hold** width/xfall is the part of the selected link that is taken from the current value for the link.

For **Modify Width**, see [24.12.4.9.1 Fixed =>to RL Modifier Types "Modify Width, Hold Xfall"](#)

For **Modify Height**, see [24.12.4.9.2 Fixed => to RL Modifier Types "Modify Height, Hold Width/Xfall"](#)

For **Modify Xfall**, see [24.12.4.9.3 Fixed => to RL Modifier Types "Modify Xfall, Hold Width"](#)

**RL** real box

*the RL (Elevation) to be reached at the end of the link.*

**Side to search** choice box left side left side, right side, both sides  
*side of the hinge string to start searching to find the string to define width/height/xfall.*

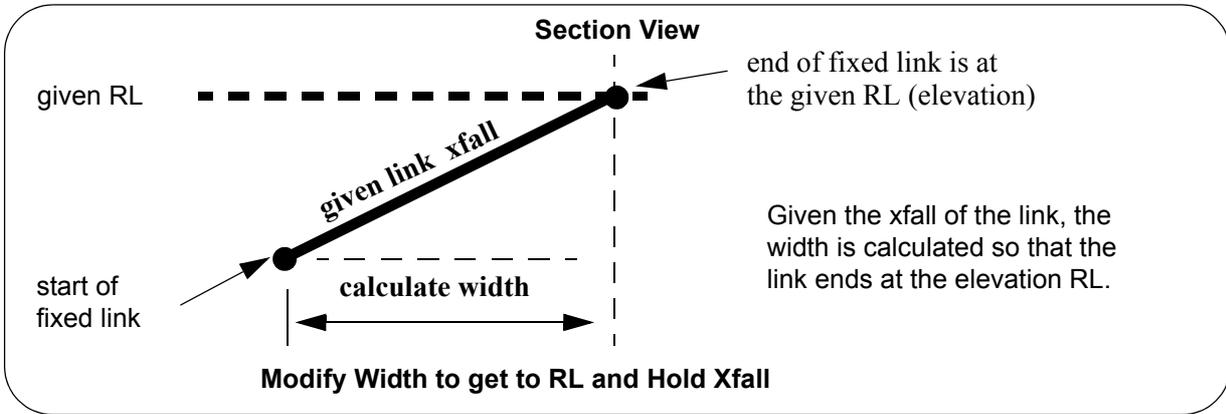
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.9.1 Fixed =>to RL Modifier Types "Modify Width, Hold Xfall"

For a fixed link **Modify Width** calculates the **width** of the link required for link to end **at the given RL**.

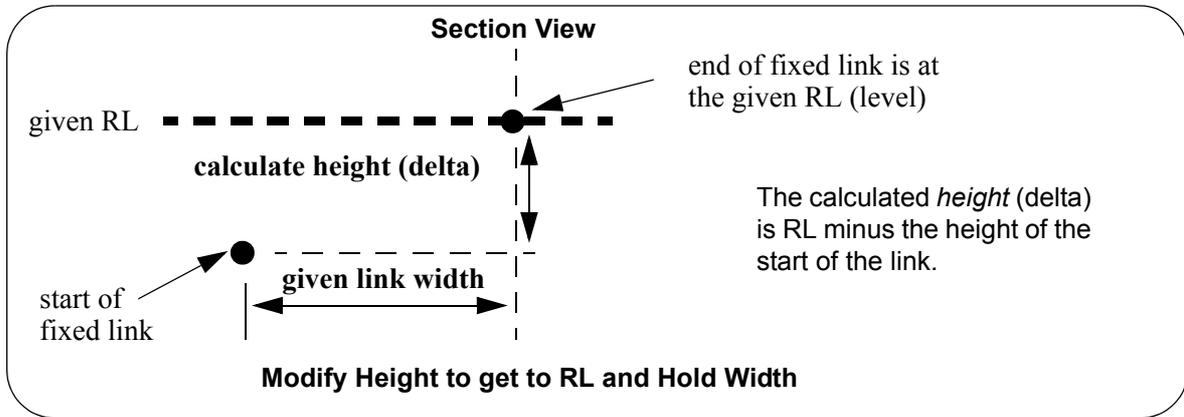
For **Modify Width, Hold Xfall** the *xfall* is taken from the link.



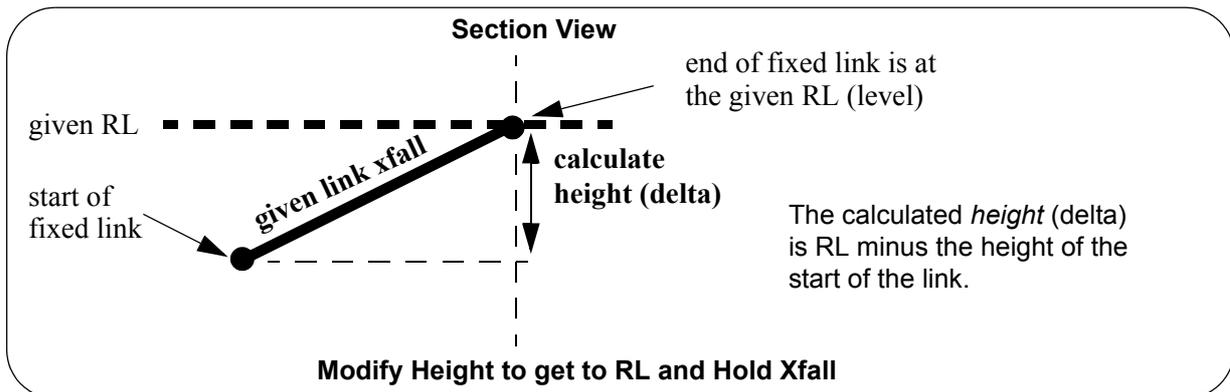
### 24.12.4.9.2 Fixed =>to RL Modifier Types "Modify Height, Hold Width/Xfall"

For a fixed link **Modify Height** calculates the **height** of the link as the difference in the height at the start point of the link, and the given **RL**.

For **Modify Height, Hold Width** the *width* is taken from the link.



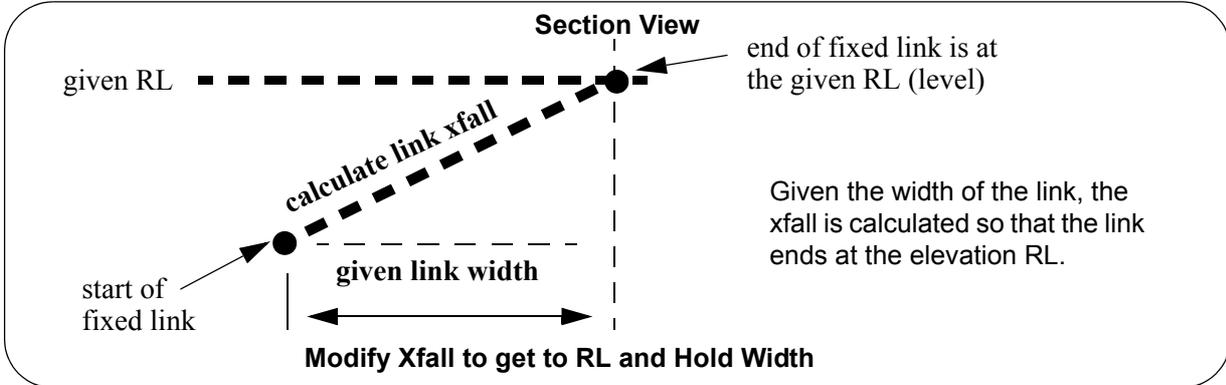
For **Modify Height, Hold Xfall** the *xfall* is taken from the link.



### 24.12.4.9.3 Fixed => to RL Modifier Types "Modify Xfall, Hold Width"

For a fixed link **Modify Xfall** calculates the **xfall** of the link as the **xfall** from the start point of the link to the **given RL**.

For **Modify Xfall, Hold Width** the *width* is taken from the link.

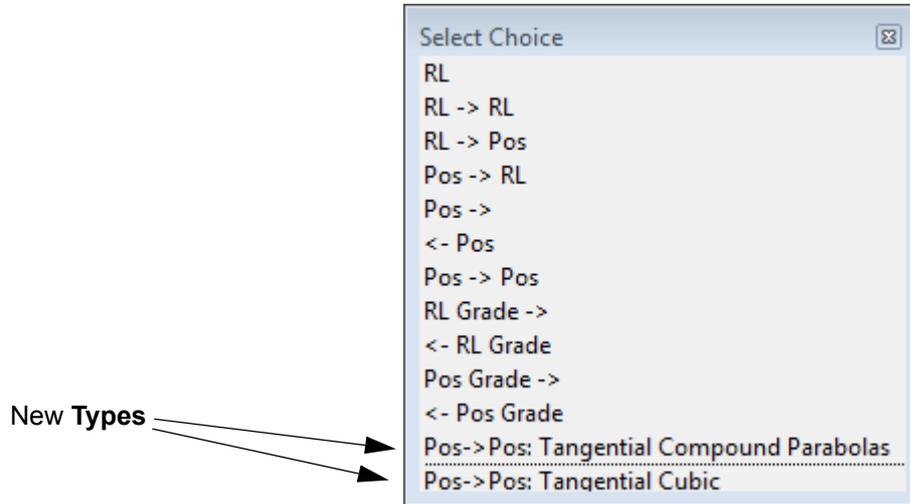


### 24.12.4.10 Left/Right Modifiers => Create =>Fixed =>to 2 Heights

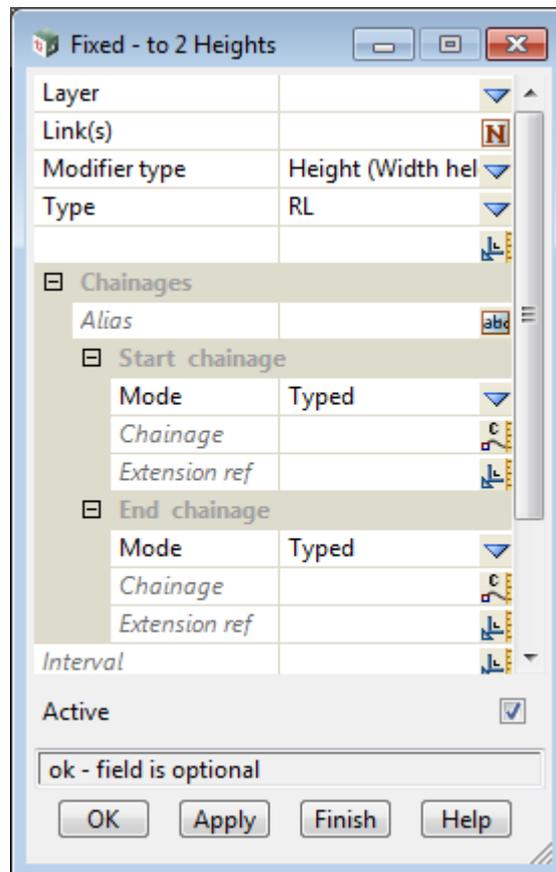
The **Fixed =>to 2 heights** option is very similar to the one in V10 except that it now has a **Layer**, **Alias**, the new **Smart Start** and **End Chainages** few more cases for **Type**.

Pos -> Pos Tangential Compound Parabolas

Pos -> Pos Tangential Cubic



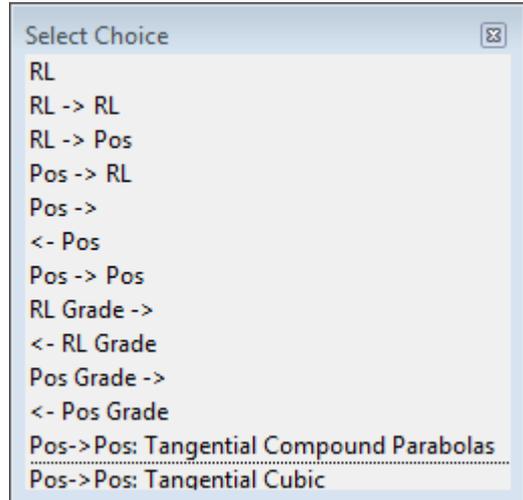
Selecting the **to 2 heights** brings up the **Fixed - to 2 Heights** panel.



The extra fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Layer** layer box Design available layers  
*name of the layer that the link to be modified is in.*  
**Type** choice box



**For Type Pos-> Pos: Tangential Compound Parabolas**

See [24.12.5.8.1.12 For Type Pos-> Pos: Tangential Compound Parabolas](#)

**For Type Pos-> Pos: Tangential Cubic**

See [24.12.5.8.1.13 For Type Pos-> Pos: Tangential Cubic](#)

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.11 Left/Right Modifiers => Create =>Fixed =>by 2 strings

The **Fixed =>by 2 strings** option calculates the width, height and xfall or slope from two given strings.

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and xfall were used to defined the fixed link, the **by 2 strings** option allows for any combination to specify how to **modify** one value to take it from between two strings whilst holding another one to what it is defined as by the link.



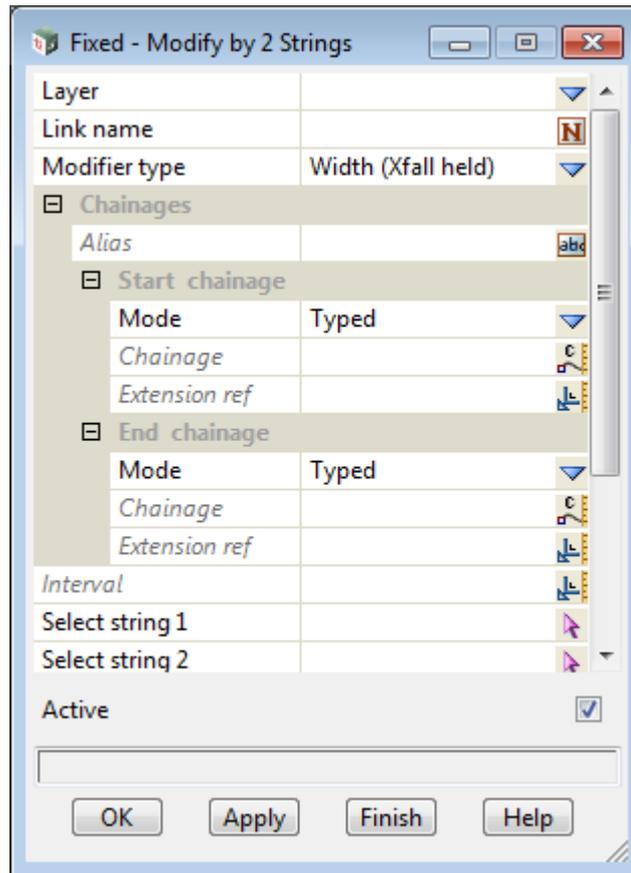
So the **by 2 strings** option replaces the V10 **Fixed** options

**Fixed =>to 2 strings =>Width to 2 strings**

**Fixed =>to 2 strings =>Height to 2 strings**

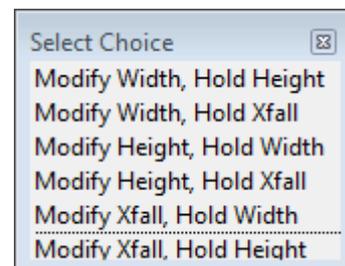
**Fixed =>to 2 strings =>Xfall to 2 strings**

Selecting the **by 2 strings** brings up the **Fixed - Modify by 2 Strings** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. This is the current link.</i>	name box		names.4d pop-up
<b>Modifier type</b>  modify Width and take Height from current link modify Width and take Xfall from current link modify Height and take Width from current link modify Height and take Xfall from current link modify Xfall and take With from current link modify Xfall and take Height from current link	choice box		



*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).*

*The **Modify** width/height/xfall is the part of the selected link that is modified from its current value by the method given by **Modifier type**.*

The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.

For **Modify Width**, see [24.12.4.11.1 Fixed =>by 2 strings Modifier Types "Modify Width, Hold Height/Xfall"](#)

For **Modify Height**, see [24.12.4.11.2 Fixed =>by 2 strings Modifier Types "Modify Height, Hold Width/Xfall"](#)

For **Modify Xfall**, see [24.12.4.11.3 Fixed => by 2 strings Modifier Types "Modify Xfall, Hold Width/Height"](#)

**String 1** string-select

*select the first string to use for defining width/height/xfall for the link.*

**String 2** string-select

*select the second string to use for defining width/height/xfall for the link.*

**Side 1 to search** choice box left side left side, right side, both sides

*side of the hinge string to start searching to find string 1 to use in defining width/height/xfall.*

**Side 2 to search** choice box left side left side, right side, both sides

*side of the hinge string to start searching to find string 2 to use in defining width/height/xfall.*

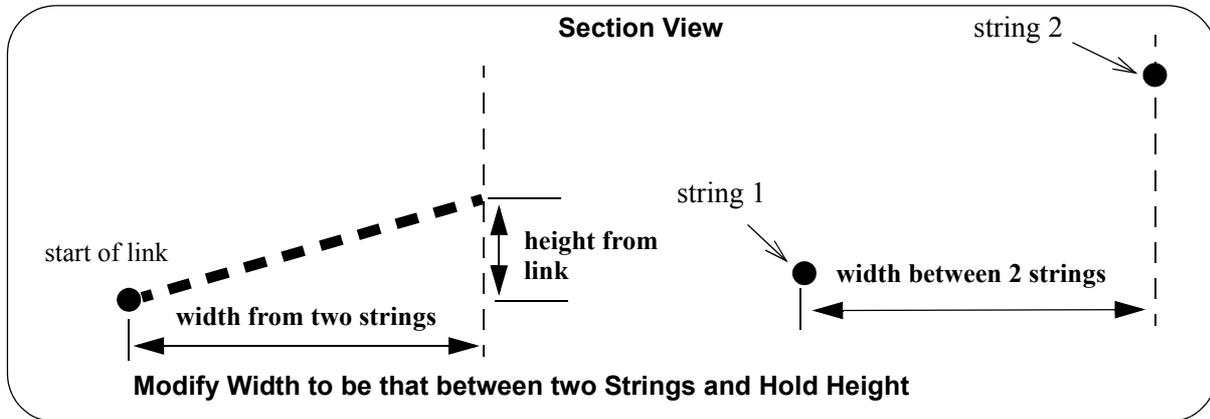
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

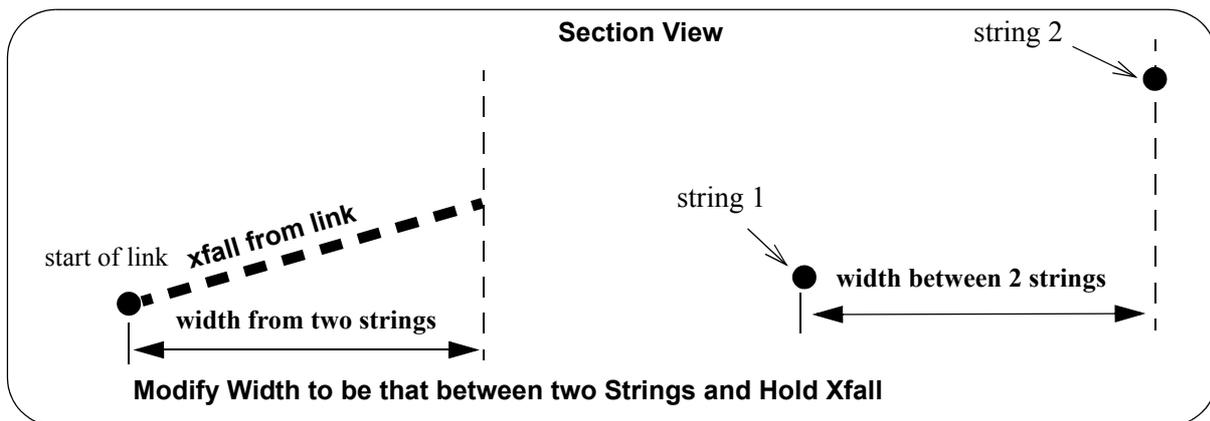
### 24.12.4.11.1 Fixed =>by 2 strings Modifier Types "Modify Width, Hold Height/Xfall"

For a fixed link **Modify Width** sets the **width** of the link to be the width between the two selected strings.

For **Modify Width, Hold Height** the *height* is taken from the link.



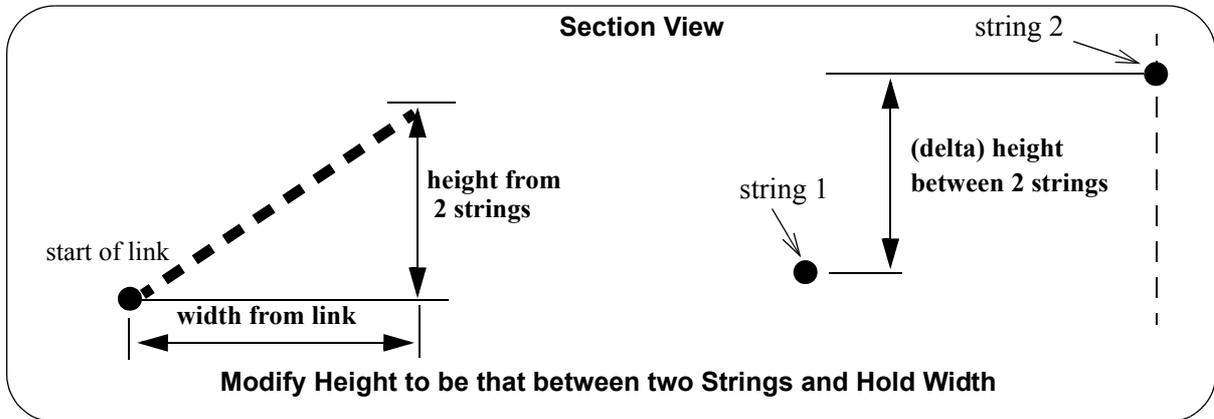
For **Modify Width, Hold Xfall** the *xfall* is taken from the link.



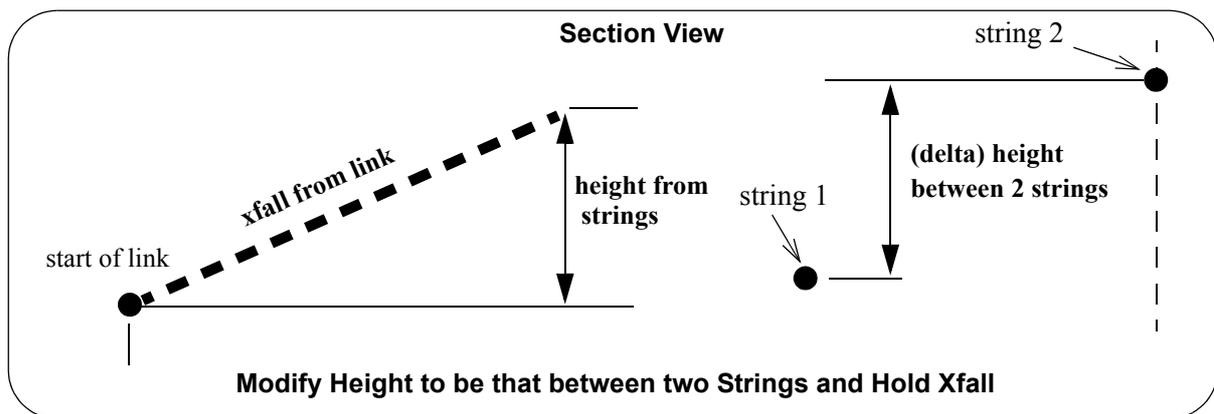
### 24.12.4.11.2 Fixed =>by 2 strings Modifier Types "Modify Height, Hold Width/Xfall"

For a fixed link **Modify Height** sets the **height** of the link to be the height between the two selected strings.

For **Modify Height, Hold Width** the *width* is taken from the link.



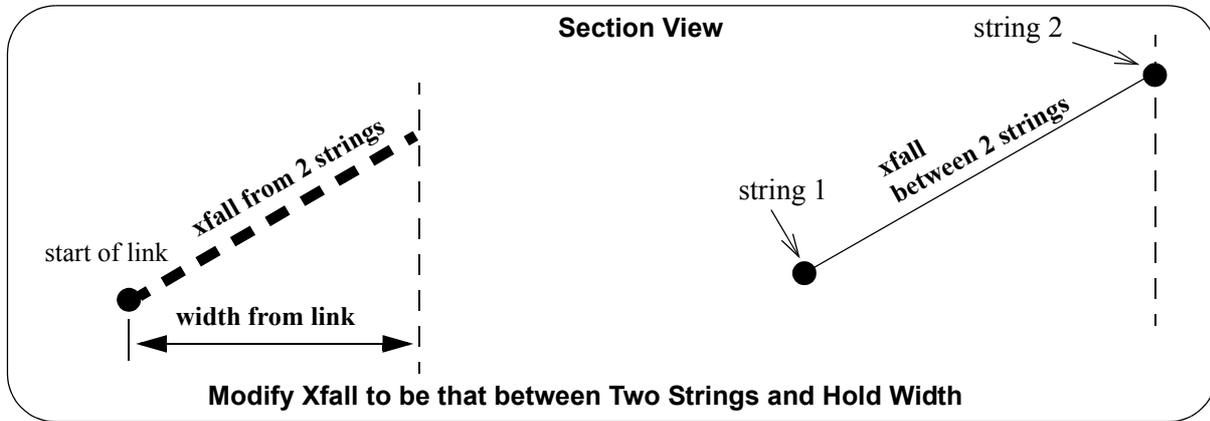
For **Modify Height, Hold Xfall** the *xfall* is taken from the link.



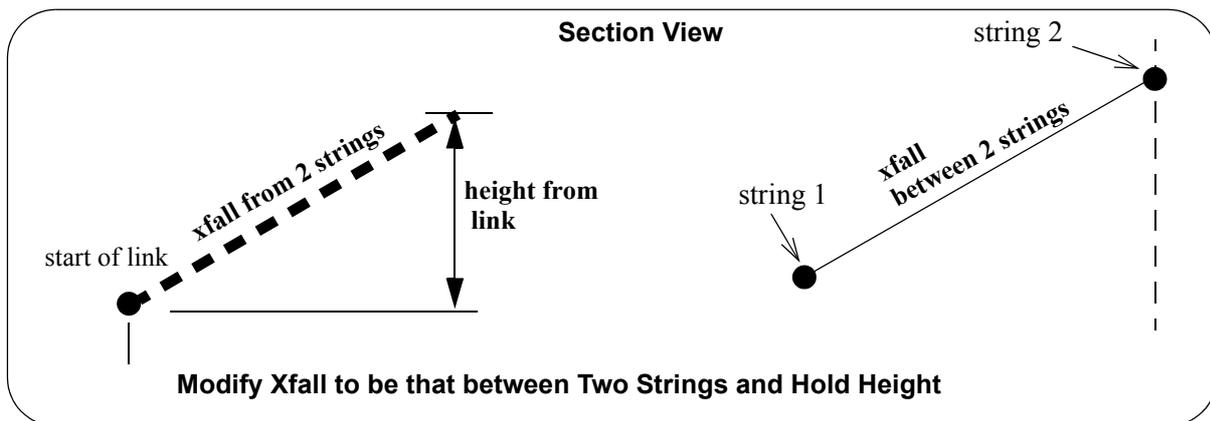
### 24.12.4.11.3 Fixed => by 2 strings Modifier Types "Modify Xfall, Hold Width/Height"

For a fixed link **Modify Xfall** sets the **xfall** of the link to be the xfall between the two selected strings.

For **Modify Xfall, Hold Width** the *width* is taken from the link.



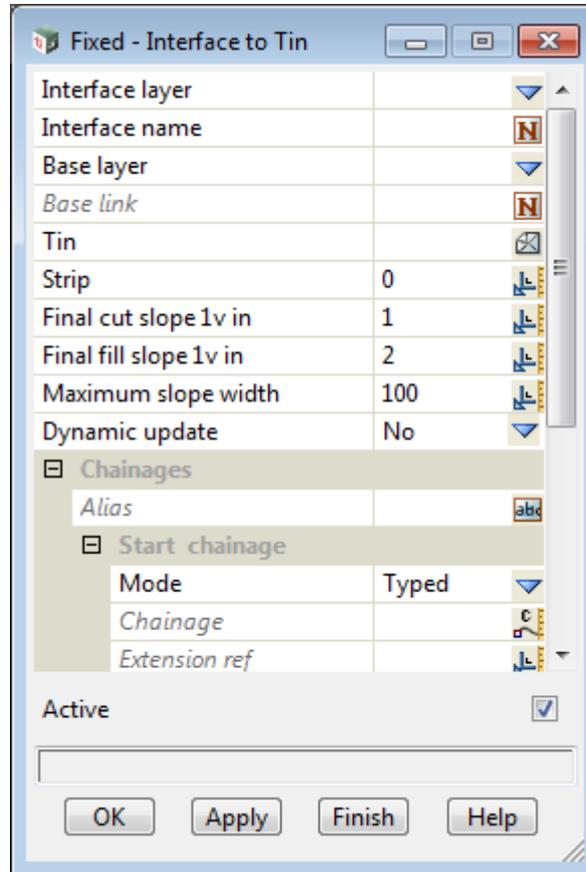
For **Modify Xfall, Hold Height** the *height* is taken from the link.



### 24.12.4.12 Left/Right Modifiers => Create =>Fixed =>Interface to tin

The **Fixed =>interface to string** option interfaces from the given link to a given tin dropped by a strip depth.

Selecting the **Interface to tin** brings up the **Fixed - Interface to tin** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to interface off is in.</i>	layer box	Design	available layers
<b>Base name</b> <i>the name of the link in the layer <b>Layer</b> to interface (batter) off.</i>	name box		names.4d pop-up
<b>Layer</b> <i>name of the layer that the created link is added to.</i>	layer box	Design	available layers
<b>Interface name</b> <i>name for the string created by this link (normally this string lies on the tin - the interface string).</i>	name box		
<b>Tin</b> <i>name of the tin to interface to.</i>	tin box		available tins
<b>Final cut slope 1 v in</b> <i>the cut slope is used if the Base link is in cut. That is, it is below the tin. The cut slope is the slope for the interface calculation to be done when the Base link is in cut. The batter continues at the cut slope until it hits the Tin dropped by the Strip depth, or until it reaches the</i>	real box	0	no slope, 0,1,2,3,4,5,10

**Maximum slope width.**

*A cut slope of one vertical to the given value of horizontal units is used. The value 0 is used to designate a **horizontal** slope - vertical slopes are not allowed.*

*For **final cut slope**, positive is up and negative down*

**Final fill slope 1 v in**      real box              0                      no slope, 0,1,2,3,4,5,10

*the fill slope is used if the Base link is in cut. That is, it is above the tin.*

*The fill slope is the slope to be used in the interface calculation when the Base link is in fill. The batter continues at the fill slope until it hits the **Tin** dropped by the **Strip** depth, or until it reaches the **Maximum slope width**.*

*A fill slope of one vertical to the given value of horizontal units is used. The value 0 is used to designate a **horizontal** slope - vertical slopes are not allowed.*

*For **final fill slope**, positive is down and negative is up.*

*This definition of fill slope being positive when going down is used so that the value in the Final fill slope 1 v in field is normally positive.*

**Maximum slope width**      real box              100

*the maximum width for the final slope. If it hasn't reached the tin by this distance then it stops.*

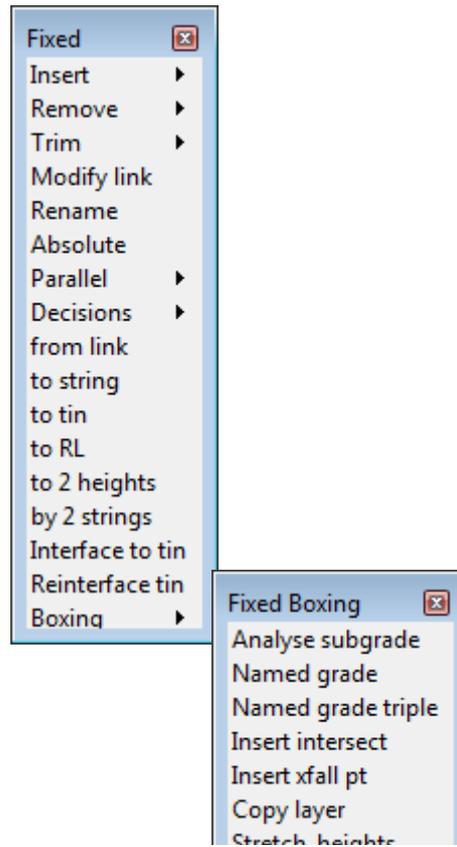
**Strip**                              real box              0

*the depth to vertically drop the tin before interfacing.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 24.12.4.13 Left/Right Modifiers =&gt; Create =&gt;Fixed =&gt;Boxing



The **Fixed =>Boxing** bring the power of the **Boxing** from **Templates** to the **Left/Right Modifiers**.

See

[24.12.4.13.1 Left/Right Modifiers => Create =>Fixed =>Boxing =>Analyse subgrade](#)

[24.12.4.13.2 Left/Right Modifiers => Create =>Fixed =>Boxing =>Named grade](#)

[24.12.4.13.3 Left/Right Modifiers => Create =>Fixed =>Boxing =>Named grade triple](#)

[24.12.4.13.4 Left/Right Modifiers => Create =>Fixed =>Boxing =>Insert intersect](#)

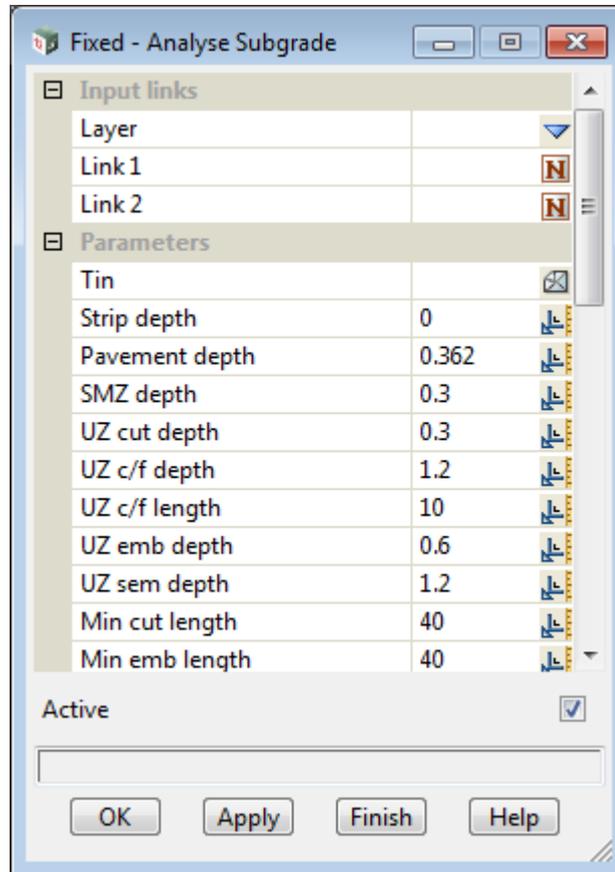
[24.12.4.13.5 Left/Right Modifiers => Create =>Fixed =>Boxing =>Insert xfall pt](#)

[24.12.4.13.6 Left/Right Modifiers => Create =>Fixed =>Boxing =>Copy layer](#)

### 24.12.4.13.1 Left/Right Modifiers => Create =>Fixed =>Boxing =>Analyse subgrade

This option is new in V11.

Selecting **Analyse subgrade** brings up the panel **Fixed - Analyse Subgrade**



**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.13.2 Left/Right Modifiers => Create =>Fixed =>Boxing =>Named grade

This option is new in V11.

Selecting **Named grade** brings up the panel **Fixed - Modify Named Grade**

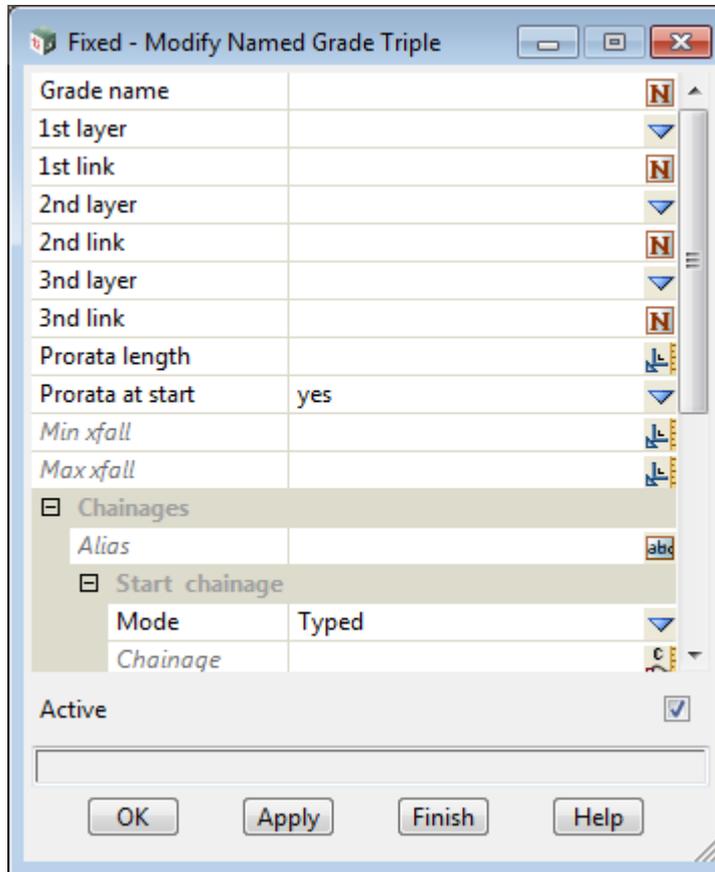
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.13.3 Left/Right Modifiers => Create =>Fixed =>Boxing =>Named grade triple

This option is new in V11.

Selecting **Named grade triple** brings up the panel **Fixed - Modify Named Grade Triple**



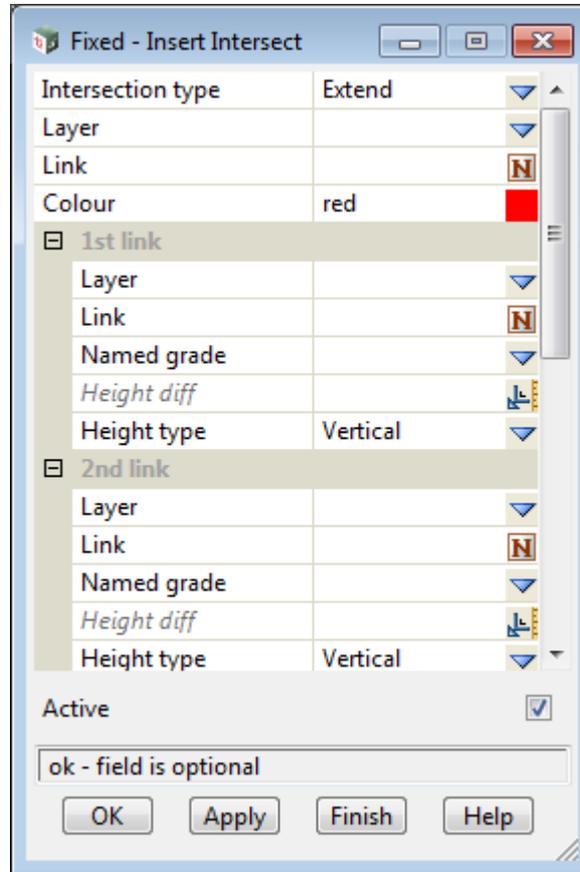
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

#### 24.12.4.13.4 Left/Right Modifiers => Create =>Fixed =>Boxing =>Insert intersect

This option is new in V11.

Selecting **Insert intersect** brings up the panel **Fixed - Insert Intersect**



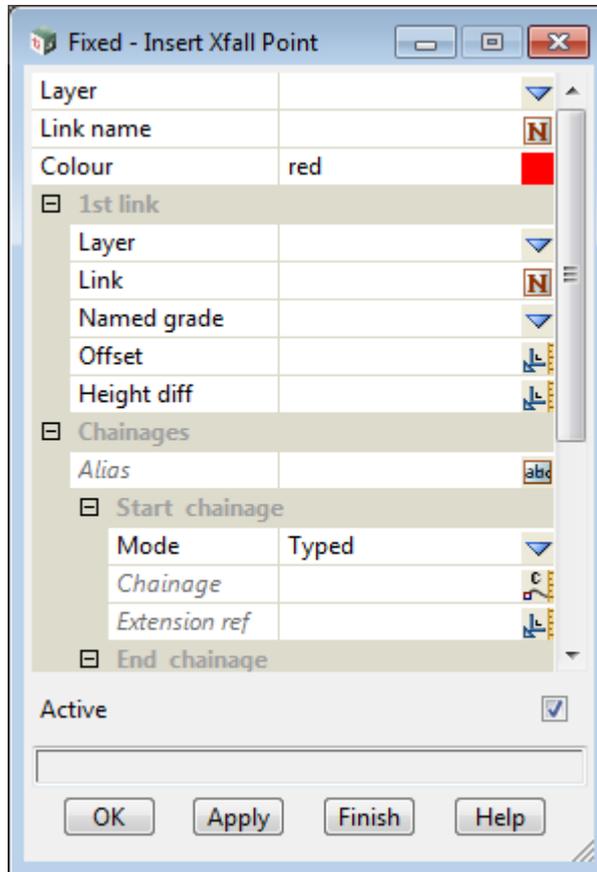
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.13.5 Left/Right Modifiers => Create =>Fixed =>Boxing =>Insert xfall pt

This option is new in V11.

Selecting **Insert xfall pt** brings up the panel **Fixed - Insert Xfall Point**



**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.4.13.6 Left/Right Modifiers => Create =>Fixed =>Boxing =>Copy layer

This option is new in V11.

The **Copy layer** command copies the **Link Points** from part of one layer to another layer, with an optional value **Height diff** added to the copied points.

That is, the **Copy layer** command copies vertices from part of a selected **Layer** (from the vertex at the end of a **Start link** to the vertex at the end of an **End link**) and copies the vertices to a given **Layer**, with an optional value **Height diff** added to the copied vertices.

The option works by

- (a) Calculating for each of the **vertices** from the vertex at the end of the **Start link** to the vertex at the end of the **End link**, the offset and height from the **Hinge Point** (the beginning of the layer).

That is, calculate an offset and delta height from the Hinge Point rather than just relative to the previous link.

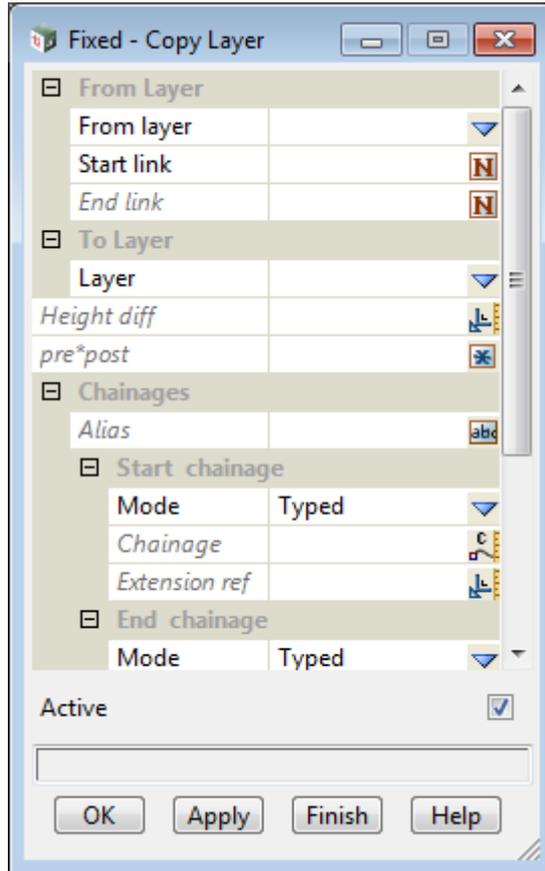
- (b) Adding the value **Height diff** to each of the height of each vertex.
- (c) Inserting the vertices into the **To Layer** using the calculated offsets and heights from the Hinge Point for each vertex.

Hence if **no links** already **exist** in the **To Layer** for the **calculated offset range** of the **copied vertices**, then the cross sectional shape of the copied links is maintained.

However if **links** already **exist** in the **To Layer** for the **calculated offset range** of the **copied vertices**, then the **existing** and **copied vertices** are **merged** together. The merging occurs by simply ordering the vertices by increasing offset and then joining adjacent vertices together. The cross sectional shape of the copied links may not be maintained.

Note that the **start vertex** of the **Start link** is not copied. This is normally not a problem unless you are trying to copy the first vertex of the first Link. To do this with the **Copy layer** command, you would need to add a new link with a width and height of zero at the beginning of the string. This new link would then be used as the **Start link**.

Selecting **Copy layer** brings up the panel **Fixed - Copy Layer**



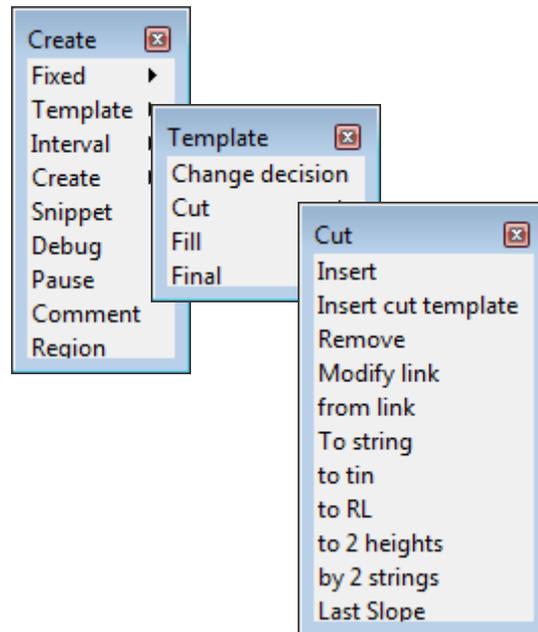
The fields and buttons used in this panel have the following functions.

Field Description	Type	Default	Pop-Up
<b>From layer</b> <i>name of the layer to copy links from.</i>	layer box	Design	available layers
<b>Start link</b> <i>name of the link in <b>From layer</b> to start copying from.</i>	link box		
<b>End link</b> <i>name of the link in <b>From layer</b> to stop copying from.</i>	link box		
<b>Layer</b> <i>name of the layer to copy links from.</i>	layer box	Design	available layers
<b>Height diff</b> <i>the value to add to the height of the vertices of the links from the <b>Start link</b> to the <b>End link</b>.</i>	real box		
<b>Pre*post</b> <i>each copied vertex is given a new name by apply the <b>Pre*pos</b> text to the original name of the vertex. That is, the text before the * is prepended to the vertex name, and the text after the * is postpended to the vertex name.</i>	text input		

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

## 24.12.5 Left/Right Modifiers => Create =>Template =>Cut



[24.12.5.1 Left/Right Modifiers Cut =>Insert .](#)

[24.12.5.2 Left/Right Modifiers Cut =>Insert cut template .](#)

[24.12.5.3 Left/Right Modifiers Cut =>Link .](#)

[24.12.5.4 Left/Right Modifiers Cut =>from link .](#)

[24.12.5.5 Left/Right Modifiers Cut =>to string .](#)

[24.12.5.6 Left/Right Modifiers Cut =>to tin .](#)

[24.12.5.7 Left/Right Modifiers Cut =>to RL .](#)

[24.12.5.8 Left/Right Modifiers Cut =>to 2 Heights .](#)

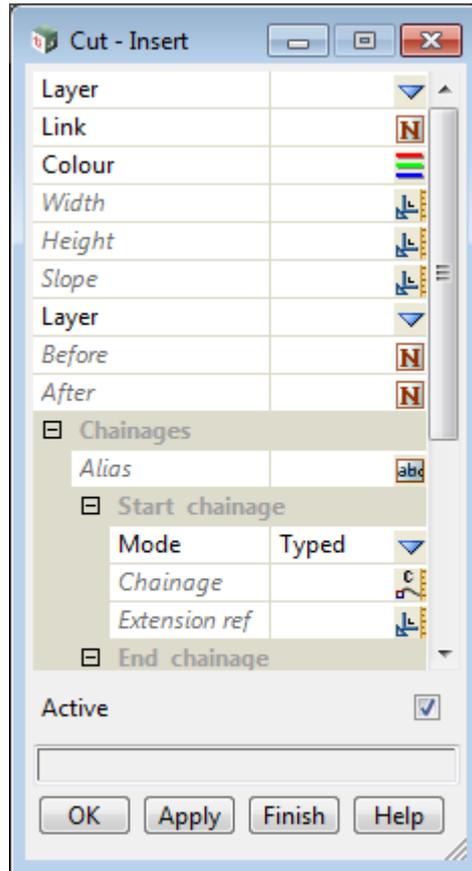
[24.12.5.9 Left/Right Modifiers Cut =>by 2 strings .](#)

*Create =>Cut =>Final Slope has not yet been documented.*

### 24.12.5.1 Left/Right Modifiers Cut =>Insert

This is similar to V10 but there is an **After** field.

Also there is a **Layer**, **Alias** and improved smart **Start** and **End** chainage modes.



Only one of **Before** or **After** can not be *blank*. If they are both blank then **Before** takes precedence over **After**.

- Layer**                                      choice box                                      available Layers  
*Layer for the new link to go into.*
- Link**                                        text box  
*name for the new cut link*
- Before**                                      choice box                                      select name menu  
*if not blank, the name of the link to insert the new link before.*  
*If blank and After is blank, the link is appended to the end of the fixed part of the template.*
- After**                                        choice box                                      select name menu  
*if Before is not blank, After is ignored.*  
*If not blank and Before is blank, the name of the link to insert the new link after.*  
*If blank and Before is blank, Before is used.*

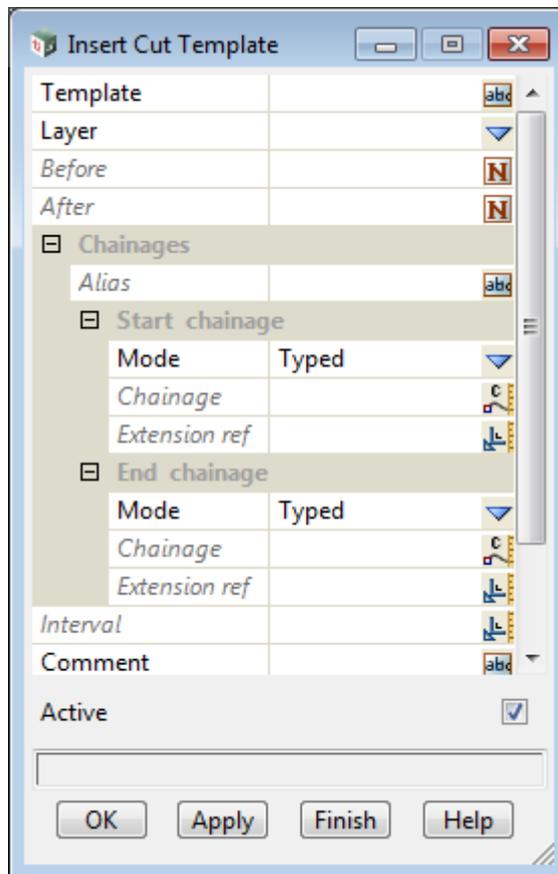
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .*

## 24.12.5.2 Left/Right Modifiers Cut =>Insert cut template

This is similar to V10 but there is an **After** field.

Also there is a **Layer**, **Alias** and improved smart **Start** and **End** chainage **modes**.



Only one of **Before** or **After** can be *not blank*. If they are both blank then **Before** takes precedence over **After**.

**Layer** choice box available Layers

*Layer for the strings in the fixed part of the template to be inserted into.*

**Before** choice box select name menu

*if **not blank**, the name of the string to insert the fixed links from the selected template before.*

*If **blank** and **After** is **blank**, the fixed links from the selected template are appended to the end of the existing fixed links.*

**After** choice box select name menu

*if **Before** is **not blank**, **After** is ignored.*

*If **not blank** and **Before** is **blank**, the name of the string to insert the fixed links from the selected template after.*

*If **blank** and **Before** is **blank**, **Before** is used.*

**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.5.3 Left/Right Modifiers Cut =>Link

The **Cut =>Link** option can modify the width, height, xfall or slope of any cut link.

For **Cut =>Link**, it doesn't matter which of the two values from width, height or slope were used to defined the fixed link, the **Link** option allows for any combination to specify how to **modify** one value of the link whilst **holding** another value of the link to be what it is currently defined as by the fixed link.



So the **Link** option replaces the V10 **Cut** options

**Cut =>Width**

**Cut =>Height**

**Cut =>Slope**

Another difference is that there are new methods for modifying the values of width, height, xfall or slope.

Modify Width	Modify Height	Modify Xfall	Modify Slope
<div style="border: 1px solid #ccc; padding: 5px;">                     Select Choice <span style="float: right;">✕</span>                      Width                      Width -&gt; Width                      Pos -&gt; Width                      Width -&gt; Pos                      Pos -&gt;                      &lt;- Pos                      Pos -&gt; Pos                      Parabola -&gt;                      &lt;- Parabola                      Circular -&gt;                      &lt;- Circular                      Cubic                      Compound Parabolas                      Reverse Curves                      Sinusoidal                      Pos -&gt; Pos : Straight line                 </div>	<div style="border: 1px solid #ccc; padding: 5px;">                     Select Choice <span style="float: right;">✕</span>                      Height                      Height -&gt; Height                      Pos -&gt; Height                      Height -&gt; Pos                      Pos -&gt;                      &lt;- Pos                      Pos -&gt; Pos                      Parabola -&gt;                      &lt;- Parabola                      Circular -&gt;                      &lt;- Circular                      Cubic                      Compound Parabolas                      Reverse Curves                      Sinusoidal                 </div>	<div style="border: 1px solid #ccc; padding: 5px;">                     Select Choice <span style="float: right;">✕</span>                      Xfall                      Xfall -&gt; Xfall                      Pos -&gt; Xfall                      Xfall -&gt; Pos                      Pos -&gt;                      &lt;- Pos                      Pos -&gt; Pos                      Parabola -&gt;                      &lt;- Parabola                      Circular -&gt;                      &lt;- Circular                      Cubic                      Compound Parabolas                      Reverse Curves                      Sinusoidal                      Xfall CRC                 </div>	<div style="border: 1px solid #ccc; padding: 5px;">                     Select Choice <span style="float: right;">✕</span>                      Slope                      Slope -&gt; Slope                      Parabola -&gt;                      &lt;- Parabola                      Circular -&gt;                      &lt;- Circular                      Cubic                      Compound Parabolas                      Reverse Curves                      Sinusoidal                      Rotate Slope as Grade                      Rotate Slope as Angle                 </div>

For example, over the chainage range, the **width** of the link can follow a parabolic, circular curve or cubic shape.

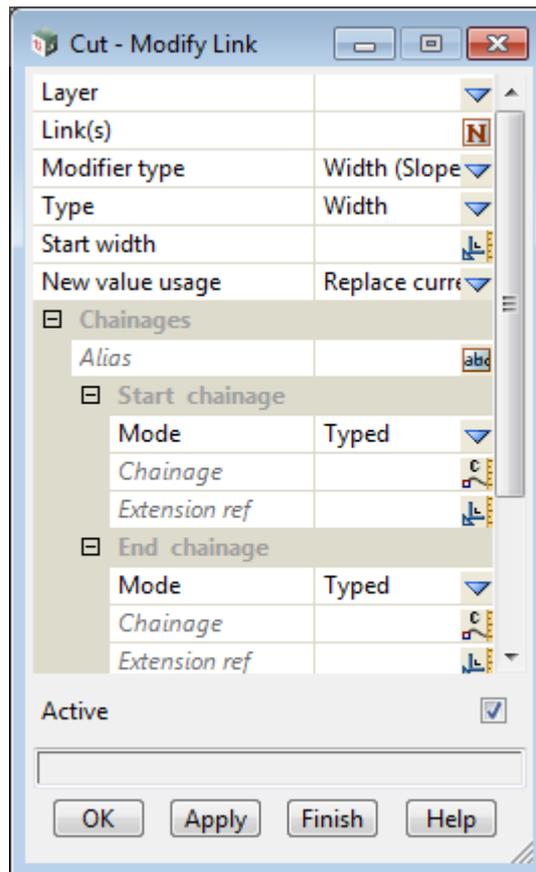
**Important Note:**

There is an important difference between Modify Xfall and Modify Slope.

**Modify Xfall** linearly interpolates the xfall between the Start Chainage and the End Chainage where xfall is the percentage of the proportion of metres vertically to metres horizontally.

**Modify Slope** linearly interpolates the slope between the Start Chainage and the End Chainage where slope is the ratio between one unit vertically to a number of units horizontally (1 in).

Another change is that **Absolute** in the *Modifier Cut => Link* command has been replaced by a choice box **New value usage** and it has been moved to just under **Modifier type**.

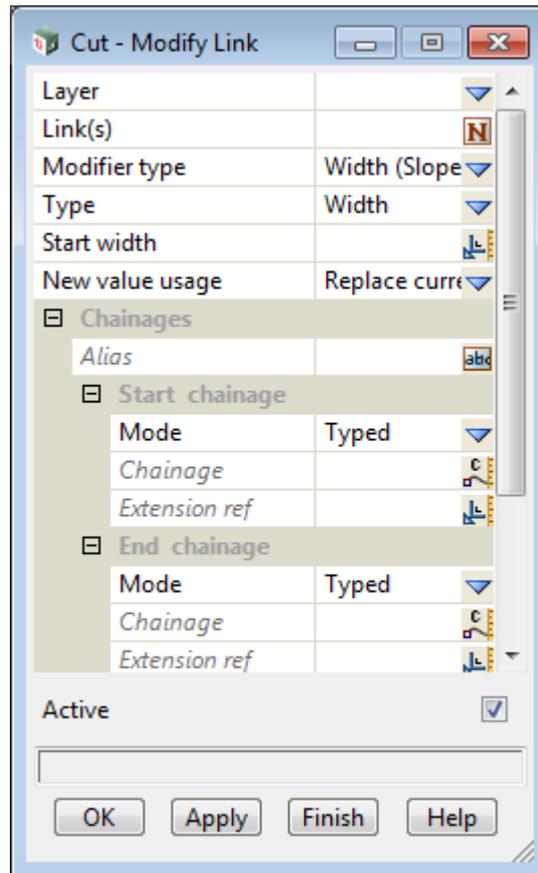


**New value usage** choice box Add to the current value of the link,  
Replace the current value of the link

*if **Add to the current value of the link**, the value of the parameter at each chainage is **added to the current value for the parameter**. This is the replacement for **Absolute not ticked**.*

*if **Replace the current value of the link**, the value of the parameter at each chainage **replaces the current value for the parameter**. This is the replacement for **Absolute ticked**.*

Selecting the **Link** brings up the **Cut - Modify Link** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>.</i>	name box		names.4d pop-up
<b>Modifier type</b>	choice box		

modify Width and take Height from selected link  
 modify Width and take Slope from selected link  
 modify Height and take Width from selected link  
 modify Height and take Slope from selected link  
 modify Xfall and take With from selected link  
 modify Xfall and take Height from selected link  
 modify Slope and take With from selected link  
 modify Slope and take Height from selected link



A link can be modified in any way regardless of the way the link was defined in terms of width, height and xfall. See [24.12.4.4.1 Type for Modifier Types "Modify Width, Hold Height/Xfall"](#)

The **Modify** width/height/xfall/slope is the part of the selected link that is modified from its current value by the method given by **Modifier type**.

For **Modify Width**, see [24.12.4.4.1 Type for Modifier Types "Modify Width, Hold Height/Xfall"](#)

For **Modify Height**, see [24.12.4.4.2 Type for Modifier Types "Modify Height, Hold Width/Xfall"](#)

For **Modify Xfall**, see [24.12.4.4.3 Type for Modifier Types "Modify Xfall, Hold Width/Height"](#)

For **Modify Slope**, see to [24.12.4.4.4 Type for Modifier Types "Modify Slope, Hold Width/Height"](#)

The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.

**New value usage** choice box Add to the current value of the link,  
Replace the current value of the link

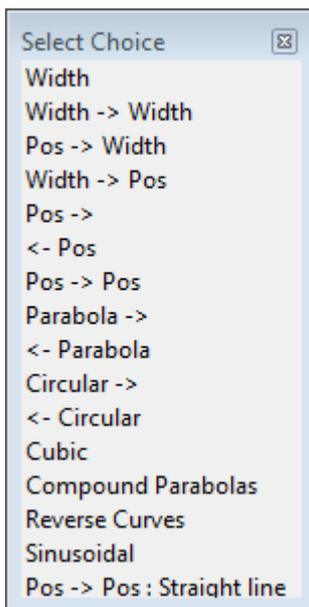
if **Add to the current value of the link**, the value of the parameter width/height/xfall/slope at each chainage is **added** to the current value for the parameter. This is the replacement for **Absolute not ticked**.

if **Replace the current value of the link**, the value of the parameter width/height/xfall/slope at each chainage **replaces** the current value for the parameter. This is the replacement for **Absolute ticked**.

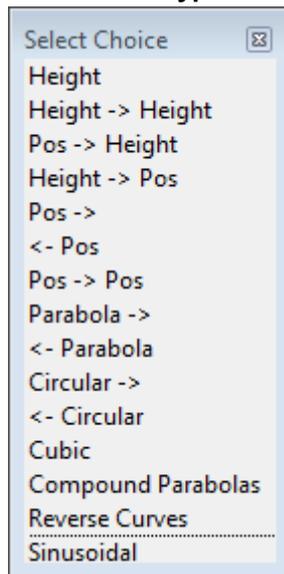
**Type** choice box

the choices for **Type** depends on the choice selected for **Modifier type**.

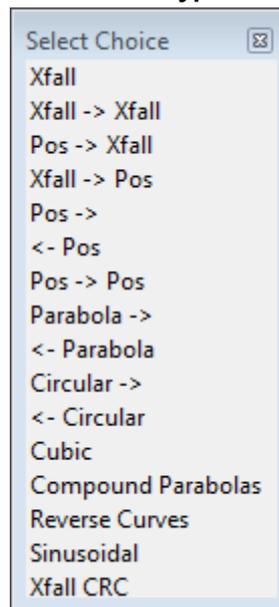
When **Modify type** is **Modify Width** the choices for **Type** are



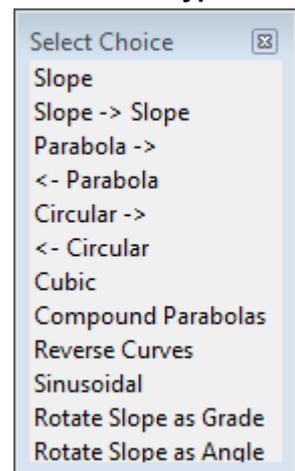
When **Modify type** is **Modify Height** the choices for **Type** are



When **Modify type** is **Modify Xfall** the choices for **Type** are



When **Modify type** is **Modify Slope** the choices for **Type** are



For **Modify Width**, see [24.12.4.4.1 Type for Modifier Types "Modify Width, Hold Height/Xfall"](#)

For **Modify Height**, see [24.12.4.4.2 Type for Modifier Types "Modify Height, Hold Width/Xfall"](#)

For **Modify Xfall**, see [24.12.4.4.3 Type for Modifier Types "Modify Xfall, Hold Width/Height"](#)

For **Modify Slope**, see [24.12.4.4.4 Type for Modifier Types "Modify Slope, Hold Width/Height"](#)

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

## 24.12.5.4 Left/Right Modifiers Cut =>from link

The **Cut =>from link** option can take the width, height and slope from another link.

It doesn't matter which of the two values from width, height or slope were used to defined the selected link, the **from link** option take one value **from** the specified link whilst holding another one to be what it is defined as by the current link.



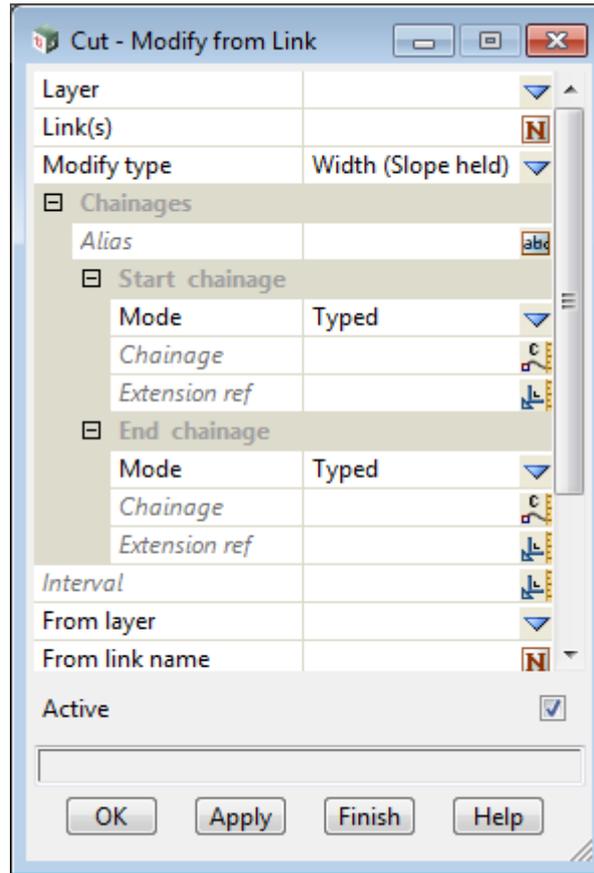
So the **from link** option replaces the V10 **Cut** options

**Cut =>from link =>Width from link**

**Cut =>from link =>Height from link**

**Cut =>from link =>Slope from link**

Selecting the **From link** brings up the **Cut - Modify from Link** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. <i>This is the current link.</i></i>	name box		names.4d pop-up
<b>Modifier type</b>	choice box		

modify Width to be as from selected link, take Height from current link  
 modify Width to be as from selected link, take Slope from current link  
 modify Height to be as from selected link, take Width from current link  
 modify Height to be as from selected link, take Slope from current link  
 modify Slope to be as from selected link, take With from current link  
 modify Slope to be as from selected link, take Height from current link



*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).*

The **Modify** width/height/xfall is the part of the selected link that is modified from its current value by the method given by **Modifier type**.

The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.

For **Modifier type Modify Width, Hold Height/Slope**, the width is taken from the link **From link name** that is in the layer **From layer** and zone **From zone**, and the **Height/Slope** is taken from the current link.

For **Modifier type Modify Height, Hold Width/Slope**, the height is taken from the link **From link name** that is in the layer **From layer** and zone **From zone**, and the **Width/Slope** is taken from the current link.

For **Modifier type Modify Slope, Hold Width/Height**, the slope is taken from the link **From link name** that is in the layer **From layer** and zone **From zone**, and the **Width/Height** is taken from the current link.

**From layer**                      layer box                      Design                      available layers  
name of the layer that the link to take values from is in.

**From Link name**                      name box    names.4d pop-up  
the name of the link in the **From Layer** and **From Zone** to get the width/height/slop from.

**From zone**                      choice box    fixed, cut, fill  
zone that the **From link name** link is in.

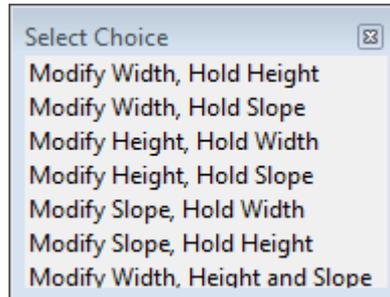
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

### 24.12.5.5 Left/Right Modifiers Cut =>to string

The **Cut =>to string** option can calculate the width, height and slop of a link by going from the start point of the link to another selected string.

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and slope were used to defined the cut link, the **to string** option allows for any combination to specify how to **modify** one value whilst holding another one to what it is defined as by the link.



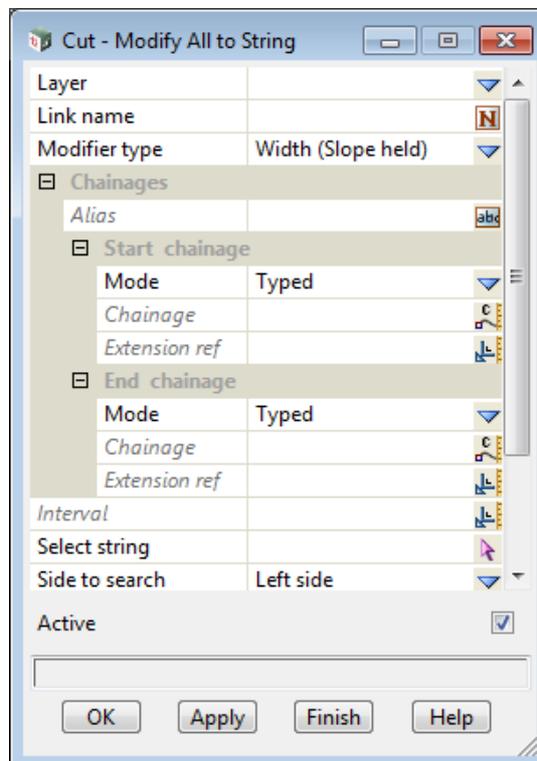
So the **to string** option replaces the V10 **Cut** options

**Cut =>to string =>Width to string**

**Cut =>to string =>Height to string**

**Cut =>to string =>Slope to string**

Selecting the **To string** brings up the **Cut - Modify All to String** panel.



The fields and buttons used in this panel have the following functions.

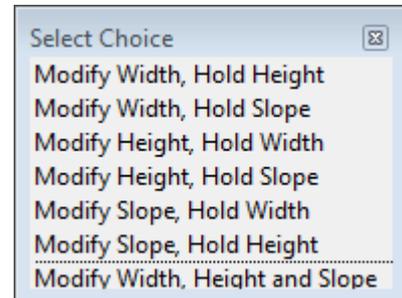
Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Layer** layer box Design available layers  
*name of the layer that the link to be modified is in.*

**Link name** name box names.4d pop-up  
*the name of the link in the Layer to modify the width/height/xfall/slope using the method given in **Modifier type**, between the chainages given by **Start mode** and **End mode**.*

**Modifier type** choice box

modify Width and take Height from selected link  
 modify Width and take Slope from selected link  
 modify Height and take Width from selected link  
 modify Height and take Slope from selected link  
 modify Slope and take With from selected link  
 modify Slope and take Height from selected link  
 modify Width, Height and Slope to get onto string



*A link can be modified in any way regardless of the way the link was defined in terms of width, height and xfall. See [24.12.4.4.1 Type for Modifier Types "Modify Width, Hold Height/Xfall"](#)*

*The **Modify** width/height/slope is the part of the selected link that is modified from its current value by the method given by **Modifier type**.*

*For **Modify Width**, see [24.12.5.5.1 Cut Modifier Types "Modify Width, Hold Height/Slope"](#)*

*For **Modify Height**, see [24.12.5.5.2 Cut Modifier Types "Modify Height, Hold Width/Slope"](#)*

*For **Modify Slope**, see [24.12.5.5.3 Cut Modifier Types "Modify Slope, Hold Width/Height"](#)*

*For **Modify Width, Height and Xfall**, see [24.12.5.5.4 Cut Modifier Type "Modify Width, Height and Slope"](#)*

*The **Hold** width/height/slope is the part of the selected link that is taken from the current value for the link.*

**Select string** string-select

*select string to use for defining width/height/slope for the link.*

**Side to search** choice box left side left side, right side, both sides

*side of the hinge string to start searching to find the string to define width/height/slope.*

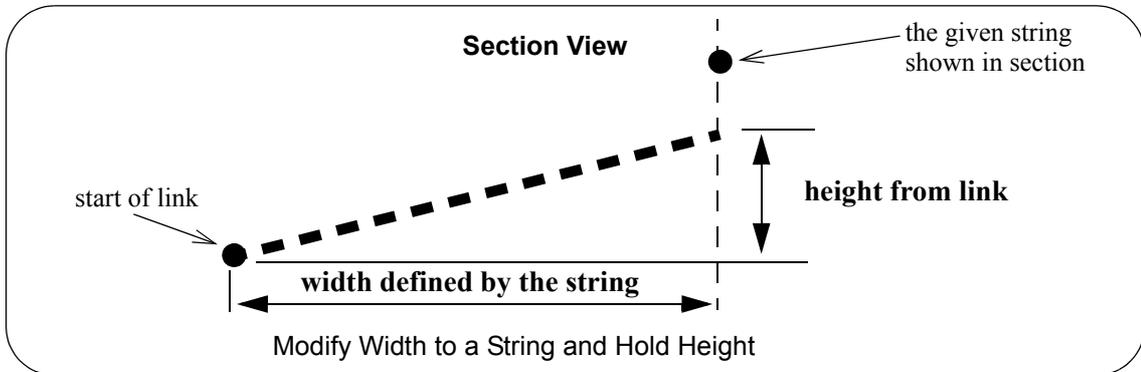
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

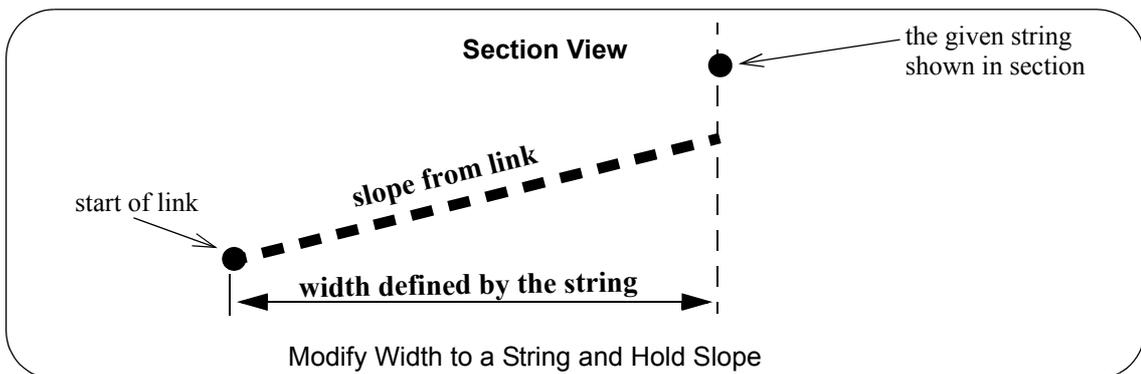
### 24.12.5.5.1 Cut Modifier Types "Modify Width, Hold Height/Slope"

For a cut link **Modify Width** calculates the **width** of the link as the width from the start point of the link, **to the selected string**.

For **Modify Width, Hold Height** the *height* is taken from the link.



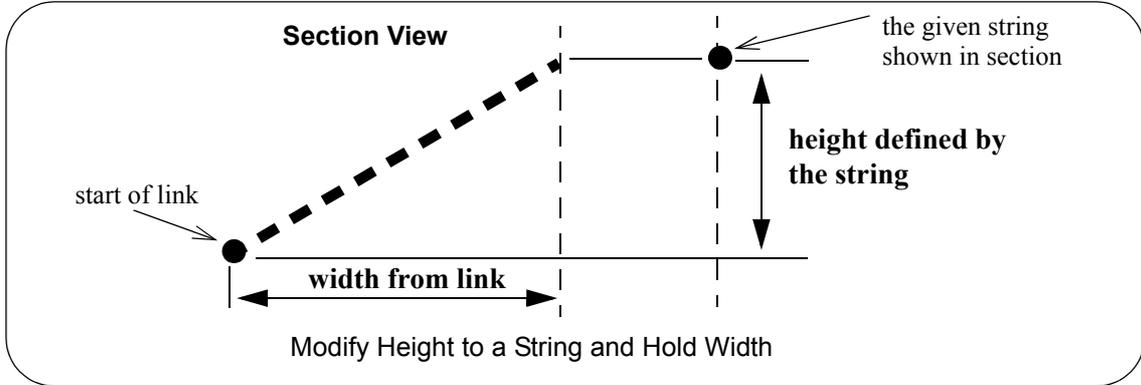
For **Modify Width, Hold Slope** the *slope* is taken from the link.



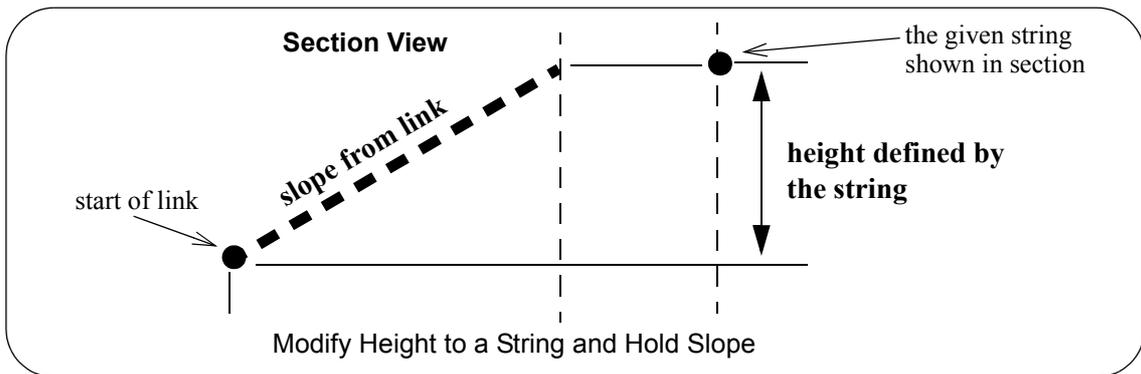
### 24.12.5.5.2 Cut Modifier Types "Modify Height, Hold Width/Slope"

For a cut link **Modify Height** calculates the **height** of the link as the difference in the height at the start point of the link, and the height **at the selected string**.

For **Modify Height, Hold Width** the *width* is taken from the link.



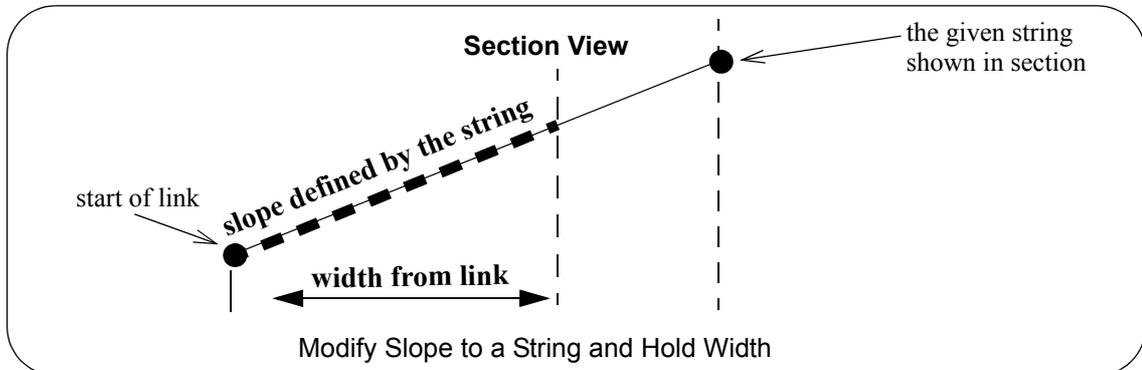
For **Modify Height, Hold Slope** the *slope* is taken from the link.



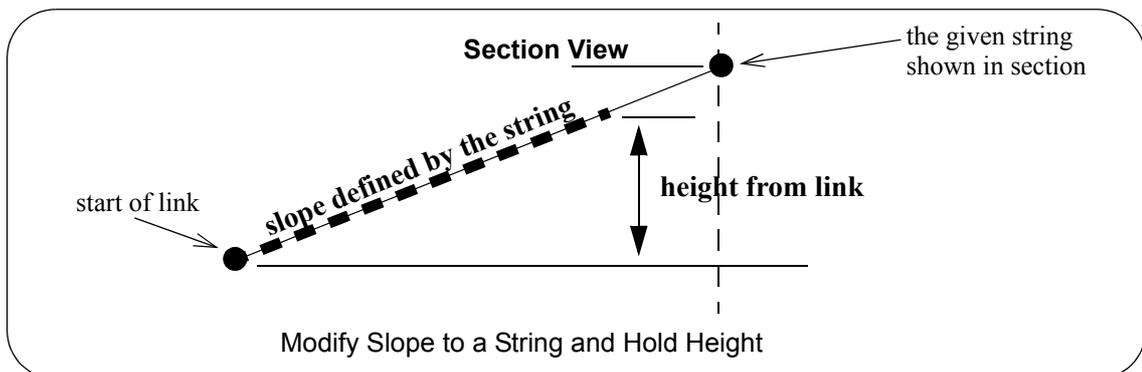
### 24.12.5.5.3 Cut Modifier Types "Modify Slope, Hold Width/Height"

For a cut link **Modify Slope** calculates the **slope** of the link as the *slope* from the start point of the link to the **selected string**.

For **Modify Xfall, Hold Width** the *width* is taken from the link.



For **Modify Slope, Hold Height** the *height* is taken from the link.



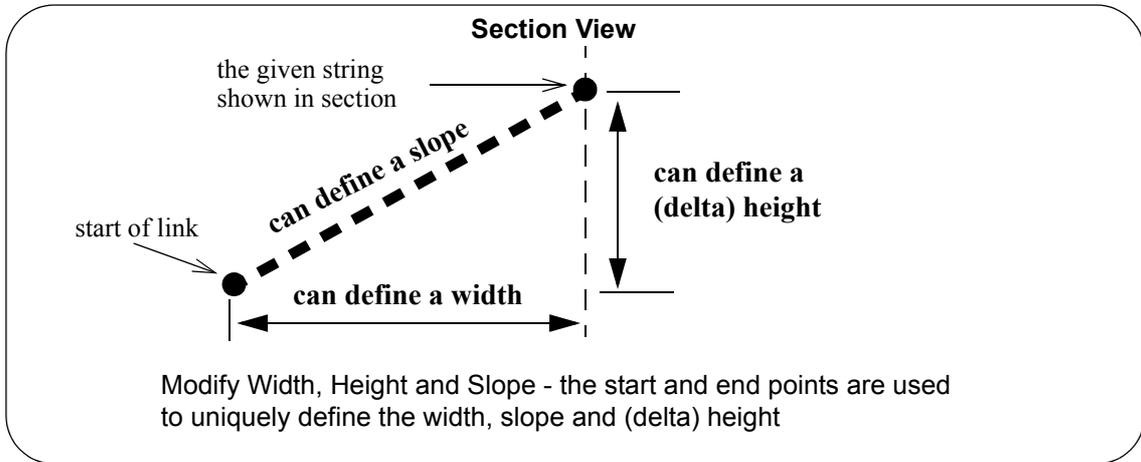
**Note:**

Using two of the above modifiers together with the same string will place the end point of the link on the selected string. For example using the same selected string, **Modify Width, Hold Slope** followed by a **Modify Slope, Hold Width** will place the end of the link on that string.

But the **To string** option with the **Modifier type Modify Width, Height and Slope** will do the same thing in one command. See [24.12.5.5.4 Cut Modifier Type "Modify Width, Height and Slope"](#).

### 24.12.5.5.4 Cut Modifier Type "Modify Width, Height and Slope"

For a cut link **Modify Width, Height and Slope** calculates the required width, height and slope of the link needed to get from the start point of the link to the **selected string**.



### 24.12.5.6 Left/Right Modifiers Cut =>to tin

The **Cut =>to tin** option calculates the width, height and xfall or slope of a link so that it is the given number intersection with a given tin. The tin does not have to be the same tin as used in the **Apply MTF** function that is using the MTF.

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and slope were used to defined the cut link, the **to tin** option allows for any combination to specify how to **modify** one value to get to the tin whilst holding another one to what it is defined as by the link. In V11 the user is also able to specify the intersection number with the tin to use rather than just taking the first as you can only do in V10.



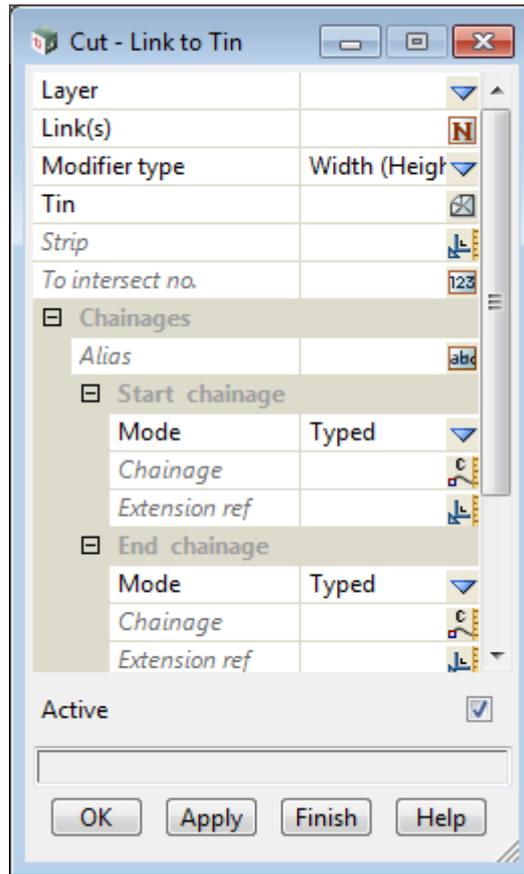
So the **to tin** option replaces the V10 **Cut** options

**Cut =>to tin =>Width to tin**

**Cut =>to tin =>Height to tin**

**Cut =>to tin =>Slope to tin**

Selecting the **To tin** brings up the **Cut - Link to Tin** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. This is the current link.</i>	name box		names.4d pop-up
<b>Modifier type</b>  modify Width and take Height from current link modify Width and take Slope from current link modify Height and take Width from current link modify Height and take Slope from current link modify Xfall and take With from current link modify Xfall and take Height from current link modify Slope and take With from current link modify Slope and take Height from current link	choice box		



A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).

The **Modify** width/height/xfall/slope is the part of the selected link that is modified from its current value by the method given by **Modifier type**.

The **Hold** width/height/xfall is the part of the selected link that is taken from the current value for the link.

For **Modify Width**, see [24.12.5.6.1 Cut =>to Tin Modifier Types "Modify Width, Hold Height/Slope"](#)

For **Modify Height**, see [24.12.5.6.2 Cut =>to Tin Modifier Types "Modify Height, Hold Width/Slope"](#)

For **Modify Xfall**, see [24.12.5.6.3 Cut => to Tin Modifier Types "Modify Xfall, Hold Width/Height"](#)

For **Modify Slope**, see [24.12.5.6.4 Cut => to Tin Modifier Types "Modify Slope, Hold Width/Height"](#)

**Tin** tin box available tins

the tin to use for defining the width/height/xfall/slope.

**To intersect no.** integer box

a tin could be intersected a number of times by the link.

If **not blank**, the number of the intersection with the tin to use. It should be greater than 0. If the number of intersections with the tin is less than this then the last intersection is used. For example, if the number is 5 then in the cases of 1,2,3,4 or 5 intersections, you get the last intersection.

If **blank**, the first intersect is used.

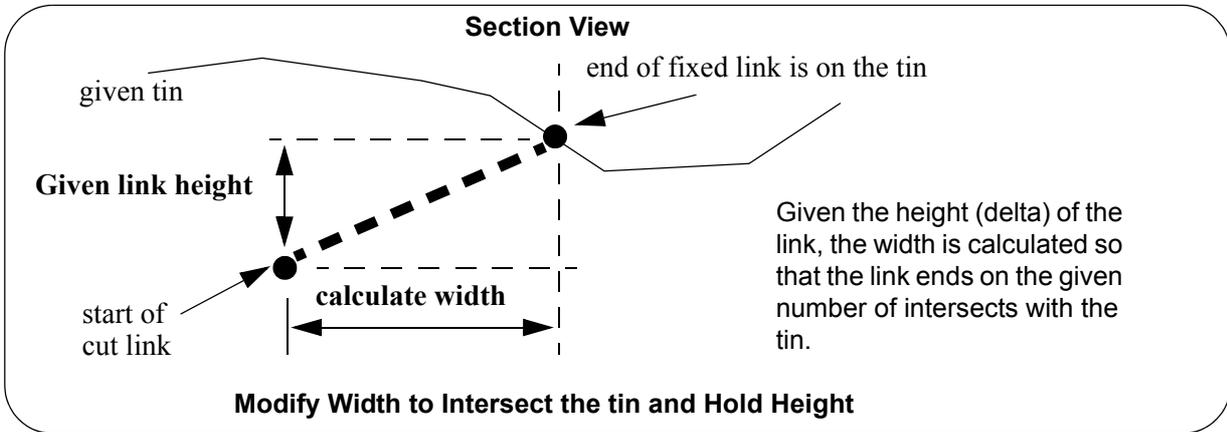
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .

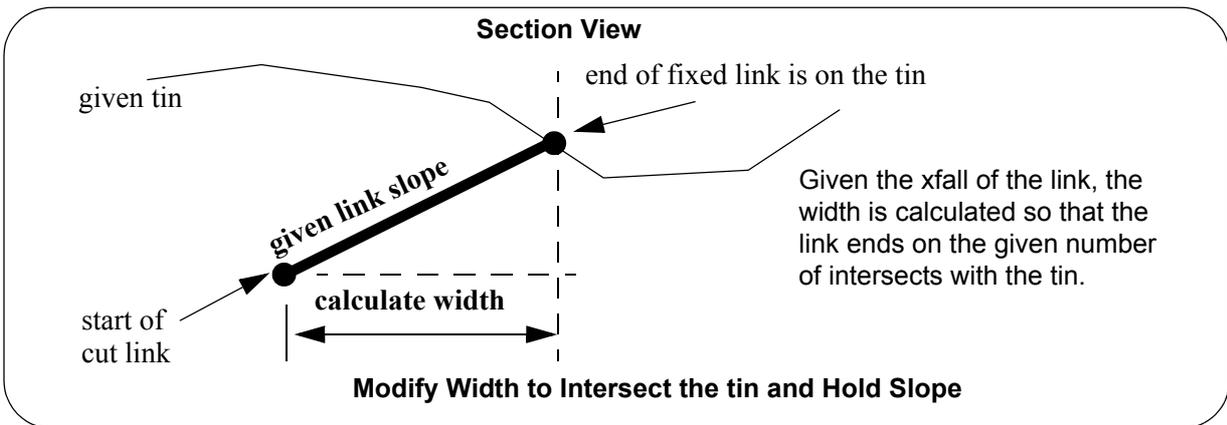
### 24.12.5.6.1 Cut =>to Tin Modifier Types "Modify Width, Hold Height/Slope"

For a cut link **Modify Width** calculates the **width** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Width, Hold Height** the *height* is taken from the link.



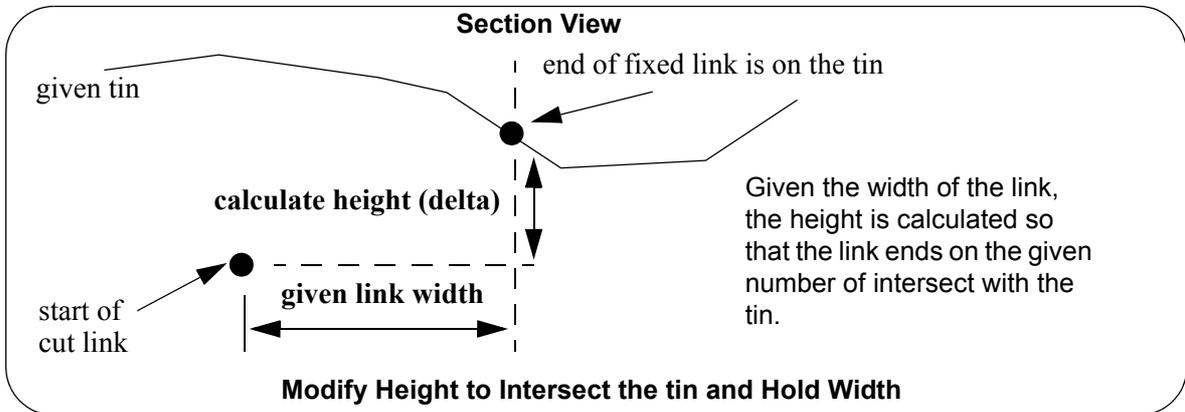
For **Modify Width, Hold Slope** the *slope* is taken from the link.



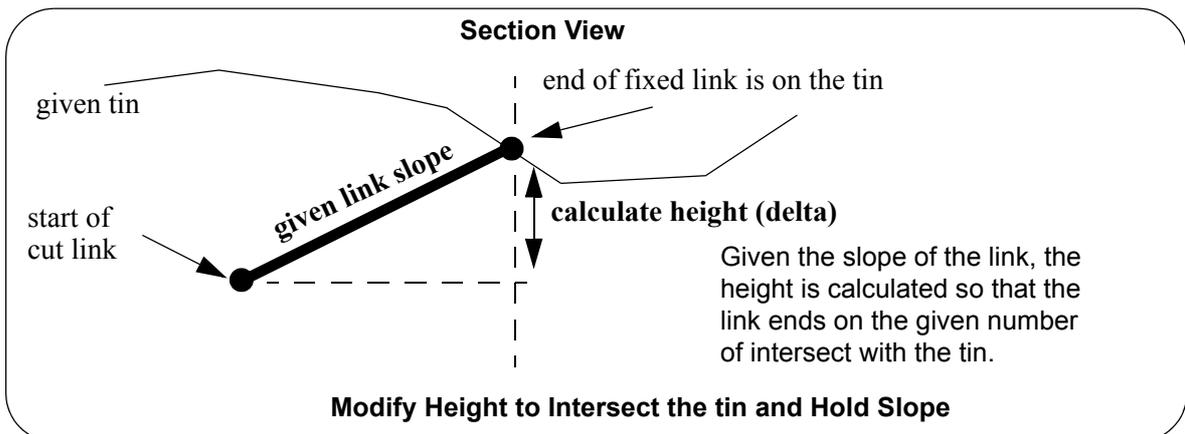
### 24.12.5.6.2 Cut =>to Tin Modifier Types "Modify Height, Hold Width/Slope"

For a cut link **Modify Height** calculates the **height** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Height, Hold Width** the *width* is taken from the link.



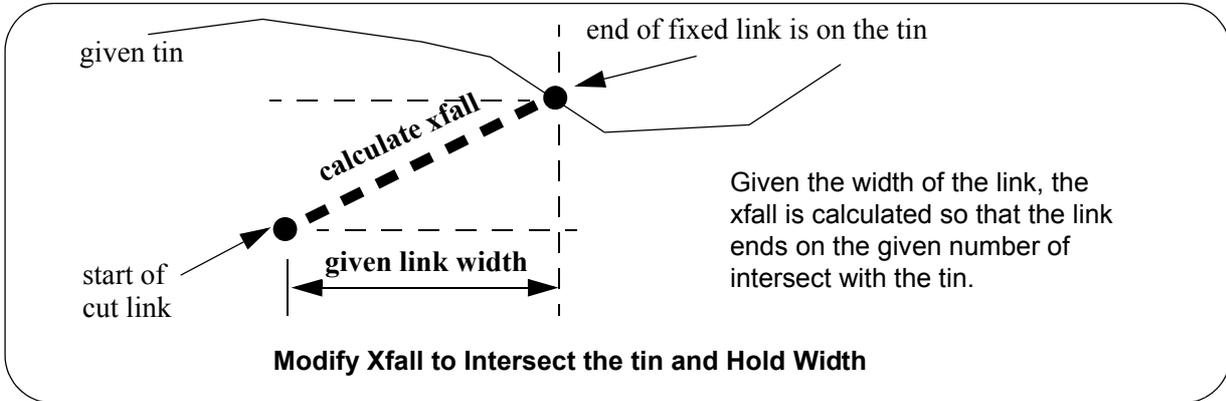
For **Modify Height, Hold Slope** the *slope* is taken from the link.



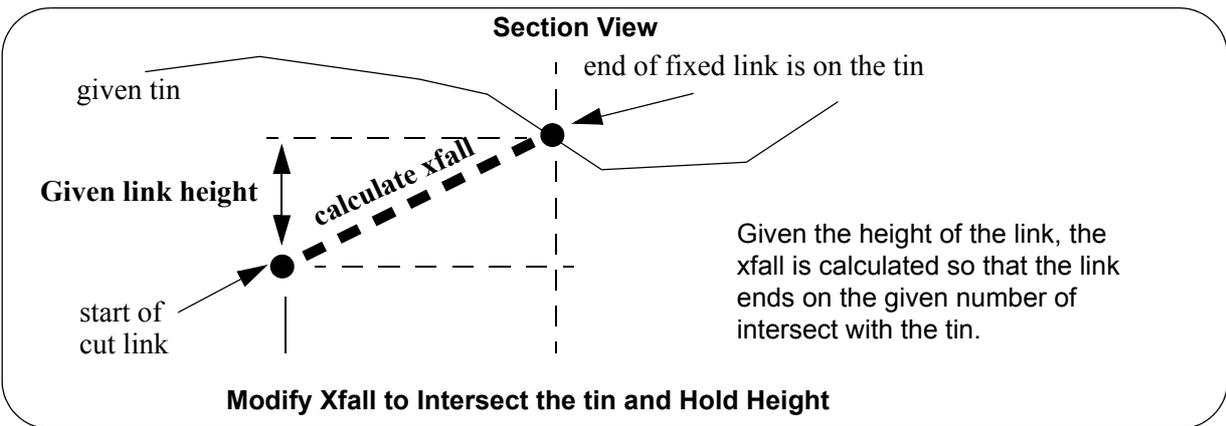
### 24.12.5.6.3 Cut => to Tin Modifier Types "Modify Xfall, Hold Width/Height"

For a cut link **Modify Xfall** calculates the **xfall** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Xfall, Hold Width** the *width* is taken from the link.



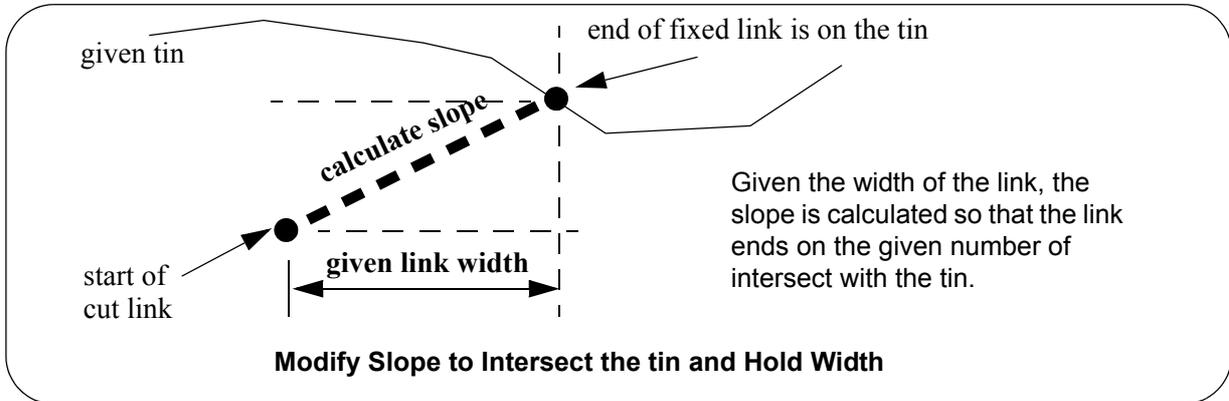
For **Modify Xfall, Hold Height** the *height* is taken from the link.



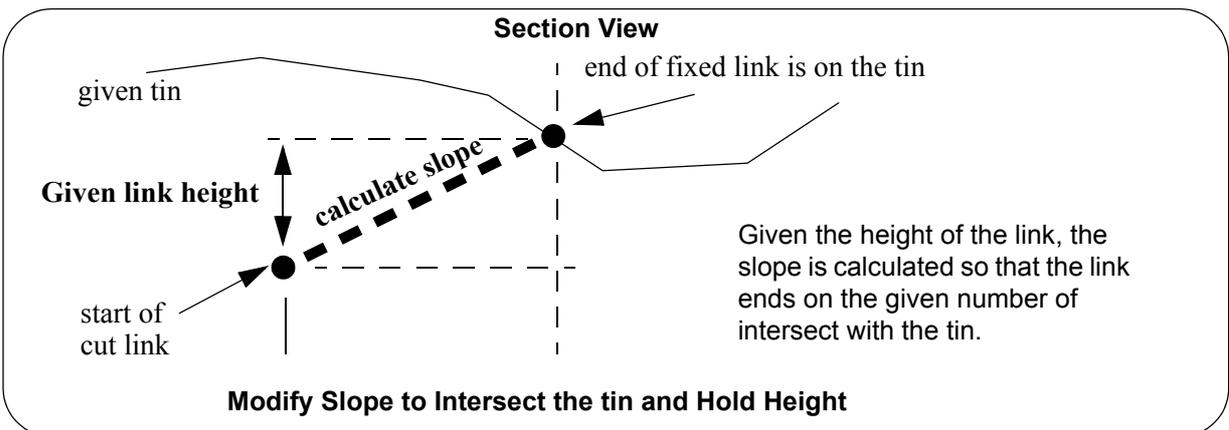
### 24.12.5.6.4 Cut => to Tin Modifier Types "Modify Slope, Hold Width/Height"

For a cut link **Modify Slope** calculates the **slope** of the link so that it intersects the tin at the number of intersects equal to **To intersect no.**

For **Modify Slope, Hold Width** the *width* is taken from the link.



For **Modify Slope, Hold Height** the *height* is taken from the link.

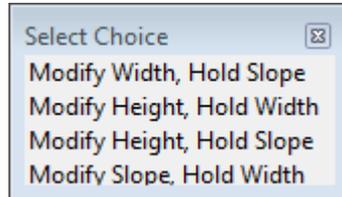


### 24.12.5.7 Left/Right Modifiers Cut =>to RL

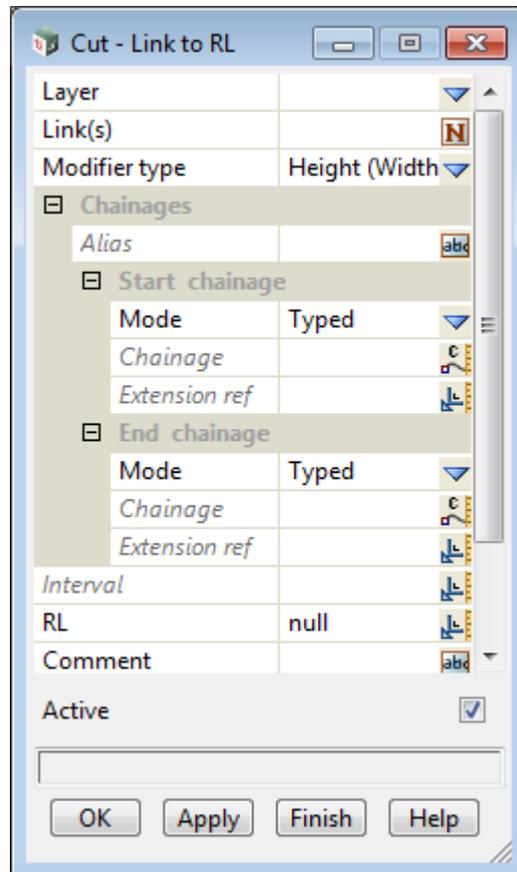
This option did not exist in V10.

The **Cut =>to RL** option can calculate the width, height and xfall of a link by going from the start point of the link to a given RL (elevation).

One restriction is of course that it makes no sense to **Hold Height**.



Selecting the **To RL** brings up the **Cut - Link to RL** panel.

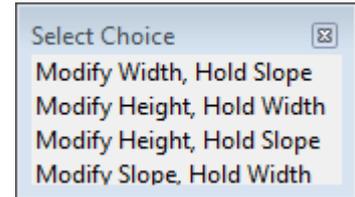


The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Layer</b> <i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b> <i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. <i>This is the current link.</i></i>	name box		names.4d pop-up

**Modifier type** choice box

modify Width and take Slope from current link  
 modify Height and take Width from current link  
 modify Height and take Slope from current link  
 modify Slope and take With from current link



*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#)*

*The **Modify** width/height/xfall is the part of the selected link that is modified from its current value by the method given by **Modifier type**.*

*The **Hold** width/xfall is the part of the selected link that is taken from the current value for the link.*

*For **Modify Width**, see [24.12.5.7.1 Cut =>to RL Modifier Types "Modify Width, Hold Slope"](#)*

*For **Modify Height**, see [24.12.5.7.2 Cut =>to RL Modifier Types "Modify Height, Hold Width/Slope"](#)*

*For **Modify Slope**, see [24.12.5.7.3 Cut => to RL Modifier Types "Modify Slope, Hold Width"](#)*

**RL** real box

*the RL (Elevation) to be reached at the end of the link.*

**Side to search** choice box left side left side, right side, both sides

*side of the hinge string to start searching to find the string to define width/height/xfall.*

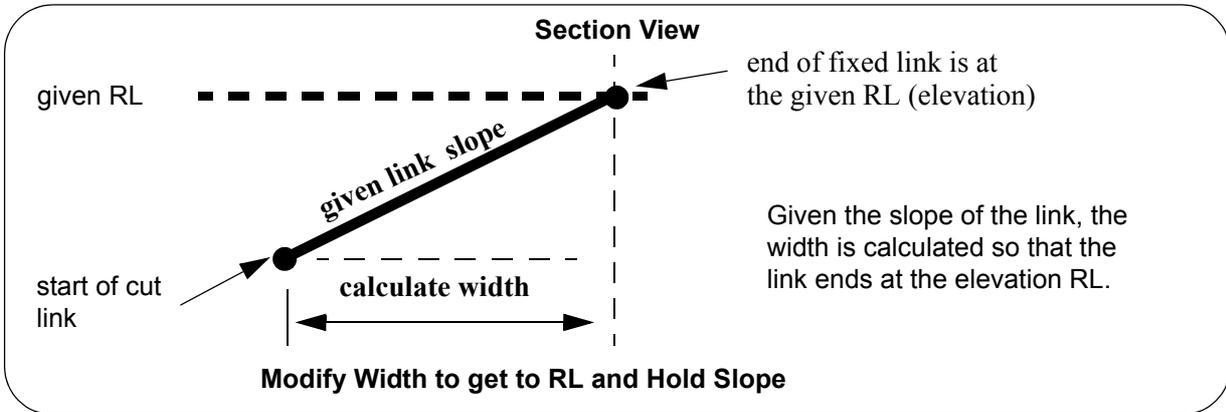
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .*

### 24.12.5.7.1 Cut =>to RL Modifier Types "Modify Width, Hold Slope"

For a cut link **Modify Width** calculates the **width** of the link required for link to end **at the given RL**.

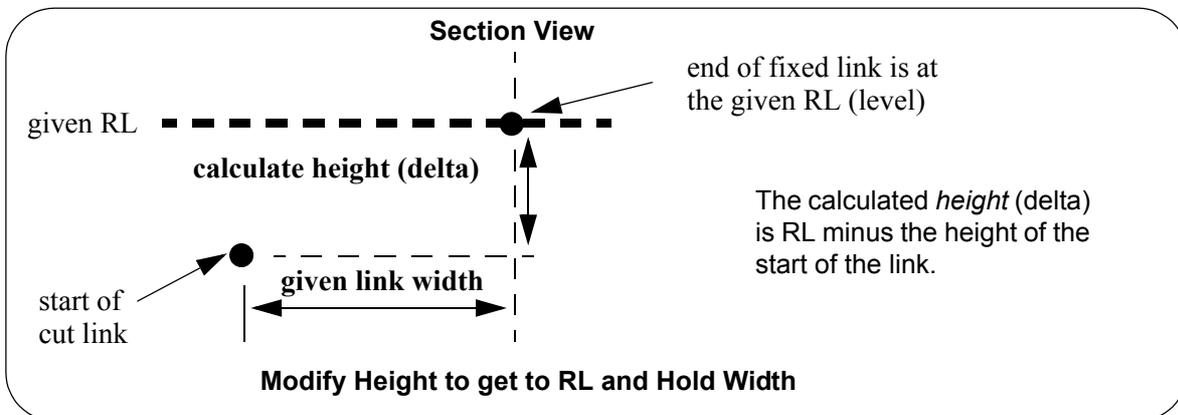
For **Modify Width, Hold Slope** the *slope* is taken from the link.



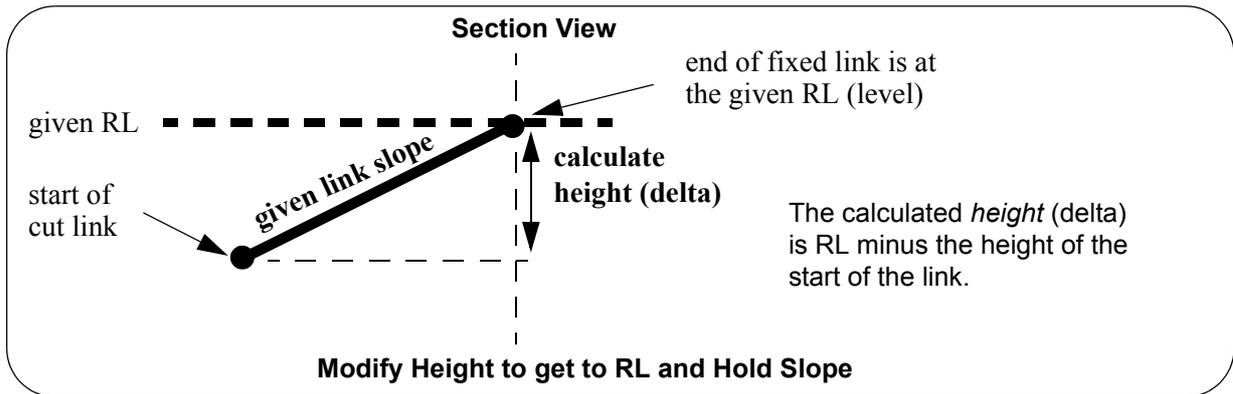
### 24.12.5.7.2 Cut =>to RL Modifier Types "Modify Height, Hold Width/Slope"

For a cut link **Modify Height** calculates the **height** of the link as the difference in the height at the start point of the link, and the given RL.

For **Modify Height, Hold Width** the *width* is taken from the link.



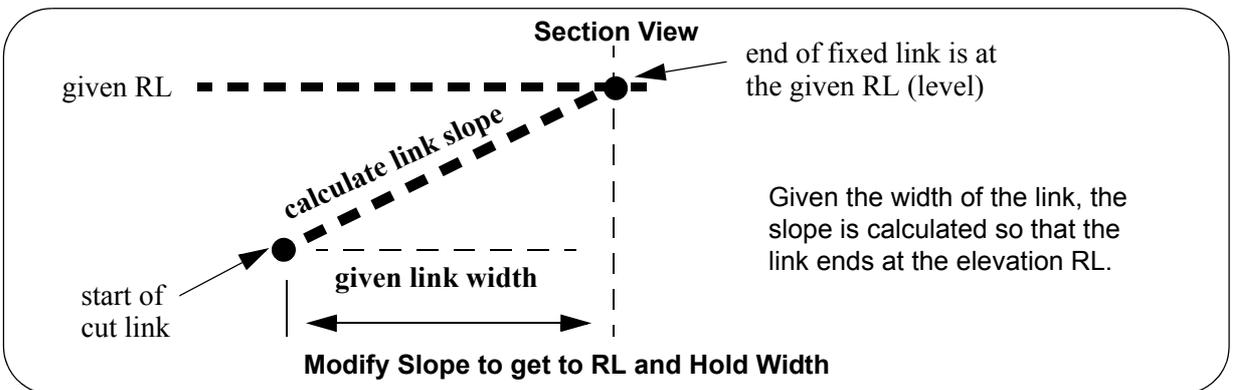
For **Modify Height, Hold Slope** the *slope* is taken from the link.



### 24.12.5.7.3 Cut => to RL Modifier Types "Modify Slope, Hold Width"

For a cut link **Modify Slope** calculates the **slope** of the link as the *slope* from the start point of the link to the **given RL**.

For **Modify Xfall, Hold Width** the *width* is taken from the link.



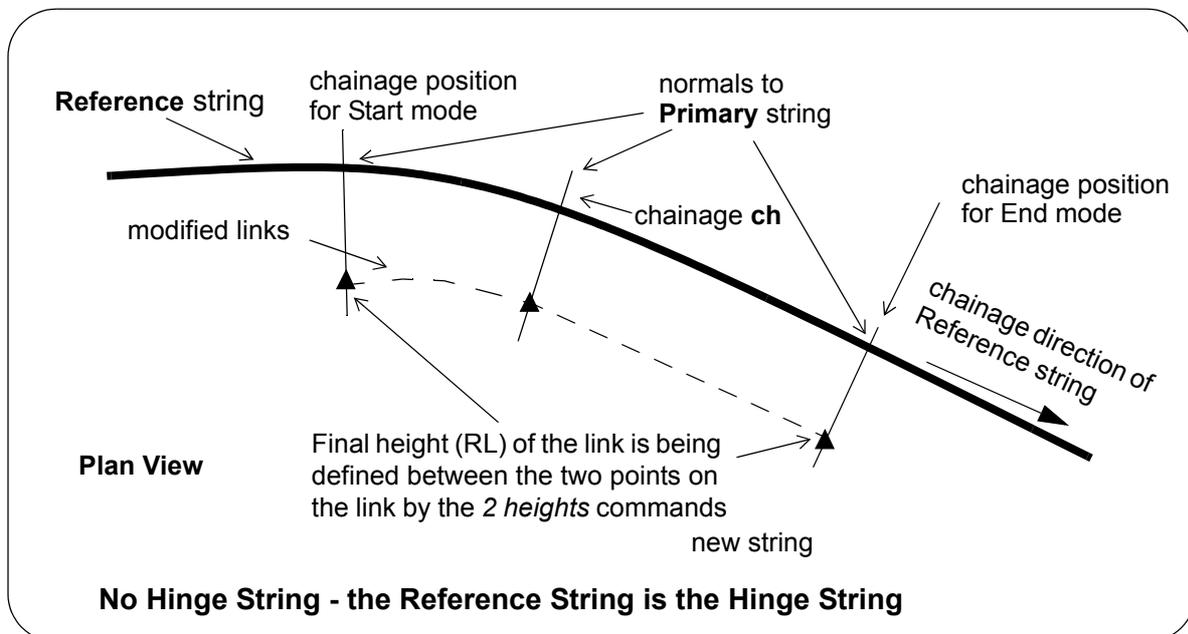
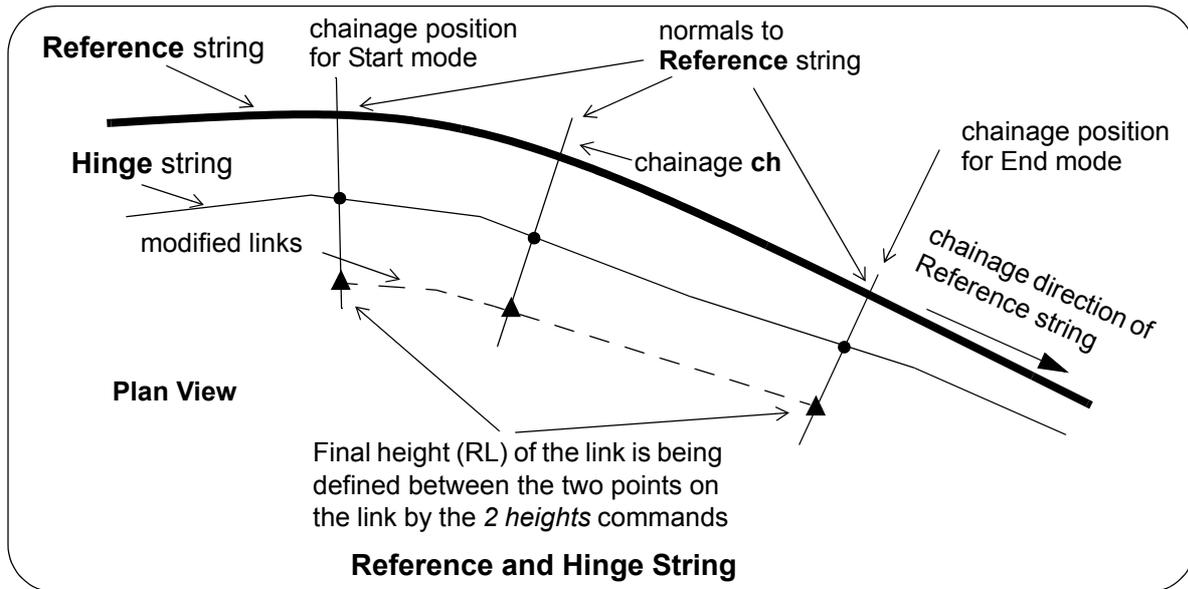
### 24.12.5.8 Left/Right Modifiers Cut =>to 2 Heights

This option is new in V11 but the calculation of Height is the same as in Fixed => 2 Heights.

However all methods will be defined here.

In this option "2 Heights" means to "2 RLs".

For an existing link, the **to 2 heights** option has a number of methods for **defining the height** from the start mode chainage to the end mode chainage. For example, over the chainage range, the height can be interpolated between two given RL's.

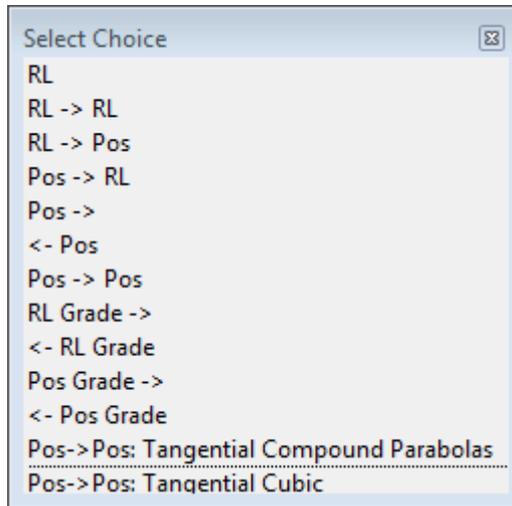


The modifier is similar to the *Width/Height/Xfall to RL* modifiers except that the RL is not constant between the vertex on the normal at *Start mode* and the vertex on the normal at *End mode*, but is determined by a given formula for the normal at each chainage value between *Start mode* and *End mode*.

In the **to 2 heights** modifiers, the formula for the height is **not** given in the chainage of the

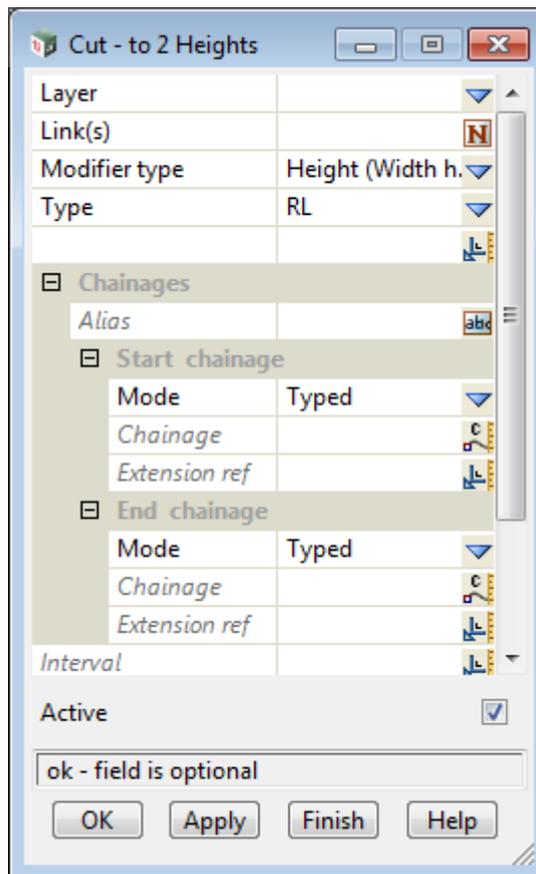
Reference string, **but in terms of the chainage along the modified string.**

The different methods for calculating the RL at each chainage are:



and for the definitions go to [24.12.5.8.1 Cut - Calculating the Heights for each Type.](#)

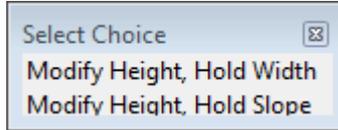
Selecting to 2 heights brings up the **Cut - to 2 Heights** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Layer</b>	layer box	Design	available layers
	<i>name of the layer that the link to be modified is in.</i>		
<b>Link name</b>	input		select name menu
	<i>name of the link to modify.</i>		
<b>Modifier type</b>	choice box		



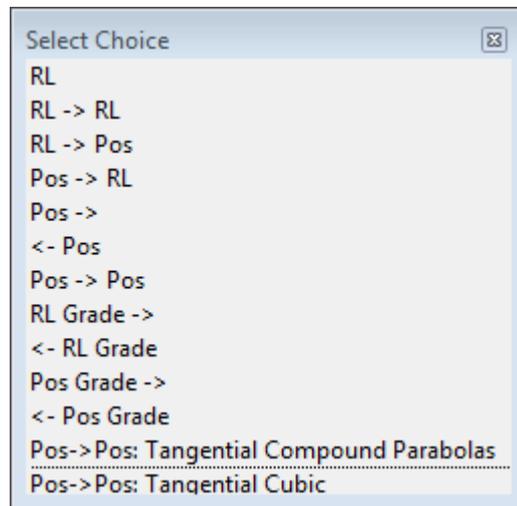
The option to **2 heights** defines the **height** of the selected string between the start and end chainages and how the height is calculated depends on the method given by **Type**.

So to completely define the link, only the width or the slope is needed.

For **Hold Width** the Width is taken from the current value of the Width of the link.

For **Hold Slope** the Slope is taken from the current value of the Slope of the link.

<b>Type</b>	choice box
-------------	------------

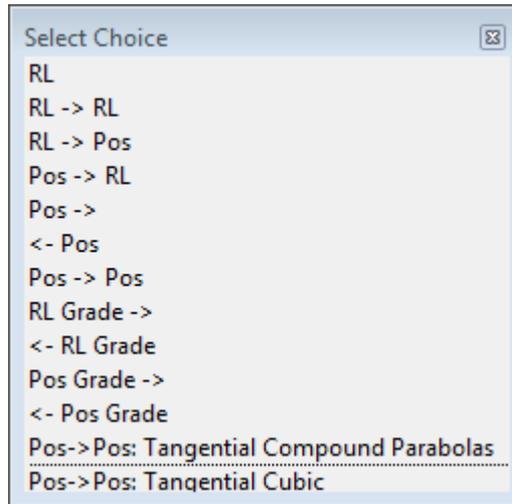


For the calculation of height for each type, go to [24.12.5.8.1 Cut - Calculating the Heights for each Type](#)

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

### 24.12.5.8.1 Cut - Calculating the Heights for each Type



for the definitions of the calculations for each choice go to

[24.12.5.8.1.1 if RL: keep a given RL](#)

[24.12.5.8.1.2 if RL -> RL: interpolate between two given RL's](#)

[24.12.5.8.1.3 if RL -> Pos: interpolate between a given start RL & the calculated end height](#)

[24.12.5.8.1.4 if Pos -> RL: interpolate between the calculated start height and a given RL at the end](#)

[24.12.5.8.1.5 if Pos ->: all points have the same height as the start point](#)

[24.12.5.8.1.6 if <- Pos: all points have the same height as the end point](#)

[24.12.5.8.1.7 if Pos-> Pos: interpolate between the calculated start and end heights](#)

[24.12.5.8.1.8 if RL Grade ->: start with a given Start RL and continue on the line at grade Grade->](#)

[24.12.5.8.1.9 if <- RL Grade: all points are on a line with given grade and going through End RL](#)

[24.12.5.8.1.10 if Pos Grade ->: start with the calculated height and continue at a given grade](#)

[24.12.5.8.1.11 if <- Pos Grade: all points are on a line of given grade and going through the end point](#)

[24.12.5.8.1.12 For Type Pos-> Pos: Tangential Compound Parabolas](#)

[24.12.5.8.1.13 For Type Pos-> Pos: Tangential Cubic](#)

#### 24.12.5.8.1.1 if RL: keep a given RL



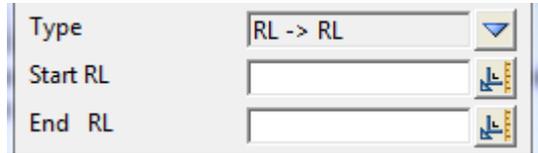
At the vertex of the link on the normal at each chainage *ch* between the **Start mode** and **End Mode**, the height at *ch* is that given in the **RL** panel field.

Hence the height is the same for all modified vertices between the *Start mode* and the *End mode*.

The modifiers of *Type RL* then work the same as the modifiers "*Width/Slope/Height to RL*" with

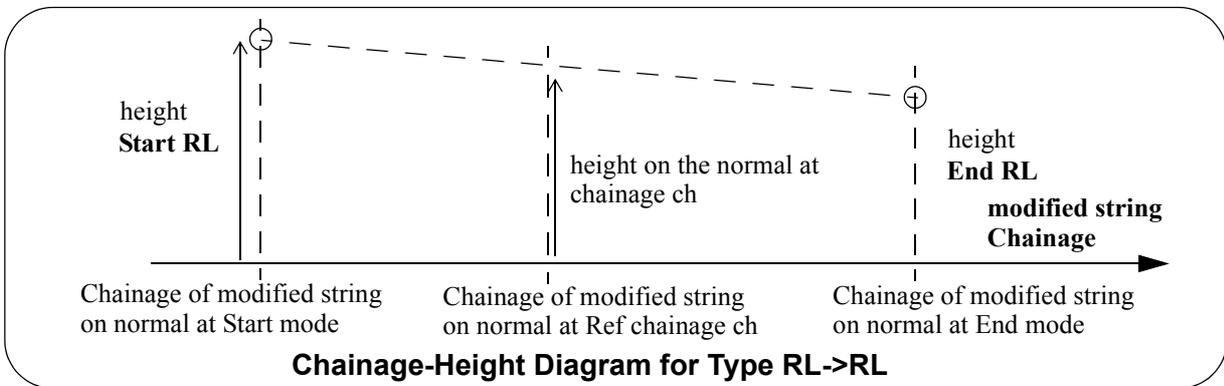
the RL being the *given RL*.

**24.12.5.8.1.2 if RL -> RL: interpolate between two given RL's**



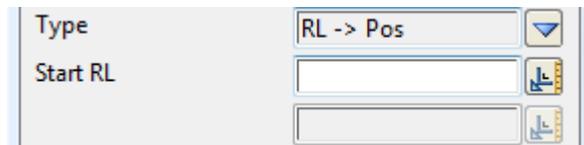
At each chainage *ch* between the **Start mode** and **End Mode**, the vertex of the modified string is on the normal to the Reference string at chainage *ch*.

The height at of the vertex on the normal to Reference chainage *ch*, is the *linear interpolation* of height with respect to the string being modified, between the given *Start RL* and the *End RL*.



Hence for the modifiers of *Type RL->RL*, the calculations at chainage *ch* are the same as the modifiers "Width/Height/Slope to RL" with the RL being the interpolated RL at chainage *ch*.

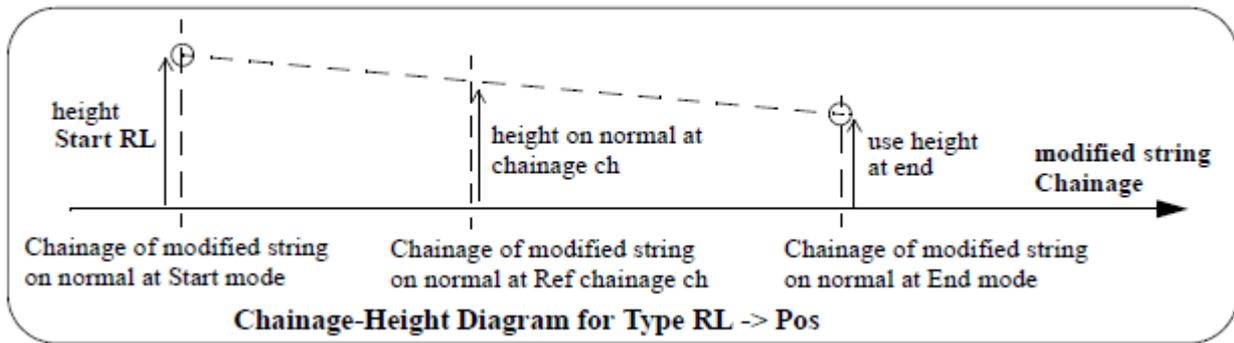
**24.12.5.8.1.3 if RL -> Pos: interpolate between a given start RL & the calculated end height**



The height at the vertex on the normal at **Start mode** is the given **Start RL**.

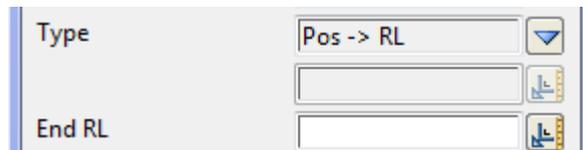
And **End RL** is the **actual** height calculated at the vertex on the modified string at the normal at **End mode** when all the modifiers before this modifier are applied.

Then at each vertex on the normals between **Start mode** and **End Mode**, the height of the vertex is the linear interpolation **with respect to modified string chainage** between the *Start RL* and the *End RL*.



Hence for the modifiers of Type *RL -> Pos*, the calculations at chainage ch are the same as the modifiers "Width/Height/Slope to RL" with the RL being the interpolated RL at chainage ch.

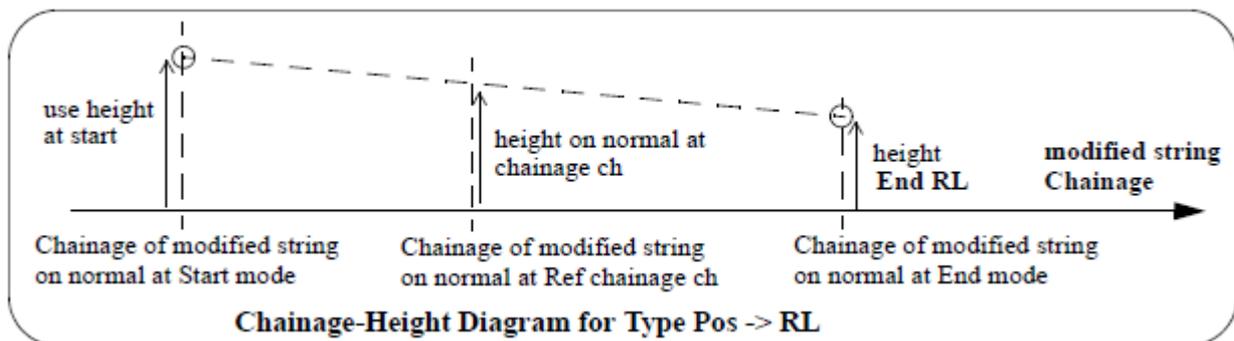
**24.12.5.8.1.4 if Pos -> RL: interpolate between the calculated start height and a given RL at the end**



**Start RL** is the **actual** height calculated at the vertex on the normal at **Start mode** when all the modifiers before this modifier are applied.

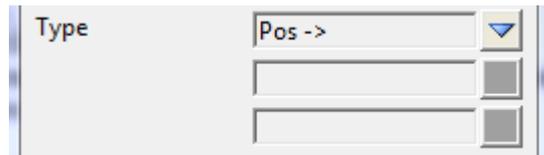
The height at the vertex on the normal at **End mode** is the given **End RL**.

Then at each vertex on the normals between **Start mode** and **End Mode**, the height of the vertex is the linear interpolation **with respect to modified string chainage** between the *Start RL* and the *End RL*.



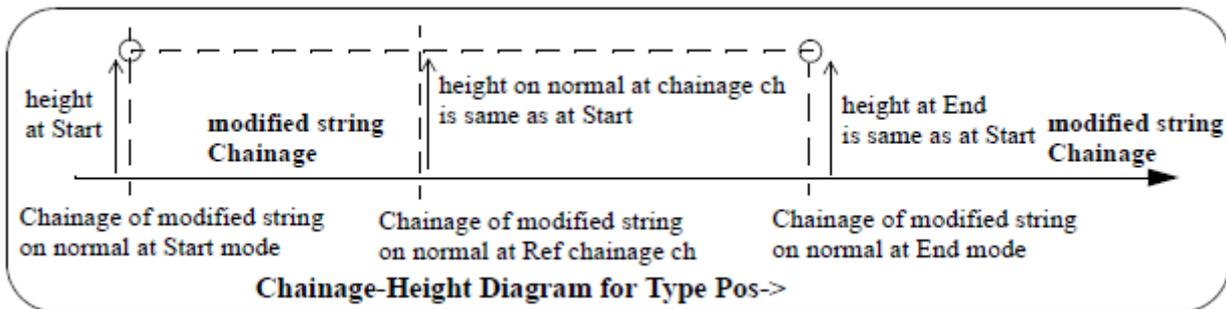
Hence for the modifiers of Type *Pos -> RL*, the calculations at chainage ch are the same as the modifiers "Width/Height/Xfall to RL" with the RL being the interpolated RL at chainage ch.

**24.12.5.8.1.5 if Pos ->: all points have the same height as the start point**



First let **Start RL** be the actual height calculated at the vertex of the string at the normal to **Start mode** when all the modifiers before this modifier are applied.

Then for all vertices on the normals between **Start mode** and **End Mode**, the height of the vertex equal to **Start RL**.



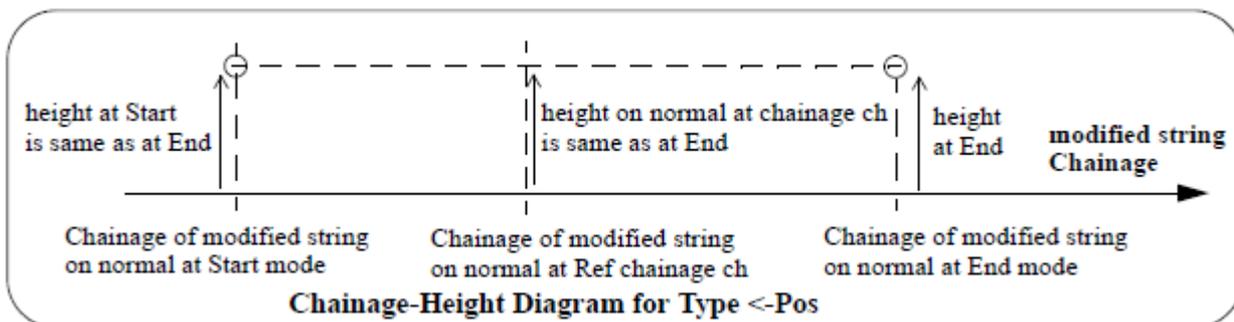
Hence the modifiers for *Type Pos->* are the same as the modifiers "Width/Slope/Height to the RL" with RL being the *Start RL*. of the link.

**24.12.5.8.1.6 if <- Pos: all points have the same height as the end point**



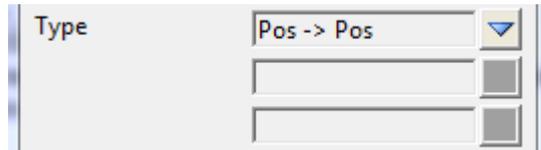
First let **End RL** be the actual height calculated at the vertex of the string at the normal to **End mode** when all the modifiers before this modifier are applied.

Then for all vertices on the normals between the **Start mode** and **End Mode**, the height of the vertex is equal to **End RL**.



Hence the modifiers for *Type <- Pos* are the same as the modifiers "Width/Xfall/Height to the RL" with RL being the *End RL*.

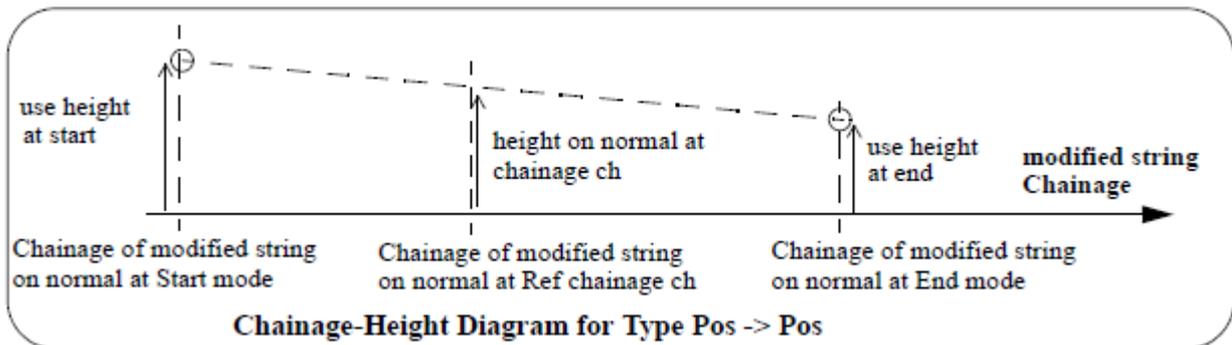
**24.12.5.8.1.7 if Pos-> Pos: interpolate between the calculated start and end heights**



First let **Start RL** be the actual height calculated at the vertex of the string at the normal to **Start mode** when all the modifiers before this modifier are applied.

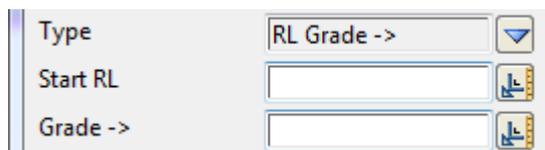
Let **End RL** be the actual height calculated at the vertex of the string at the normal to **End mode** when all the modifiers before this modifier are applied.

Then for each vertex on the normals between **Start mode** and **End Mode**, the height of the vertex is the linear interpolation **with respect to modified string chainage** between the *Start RL* and the *End RL*.



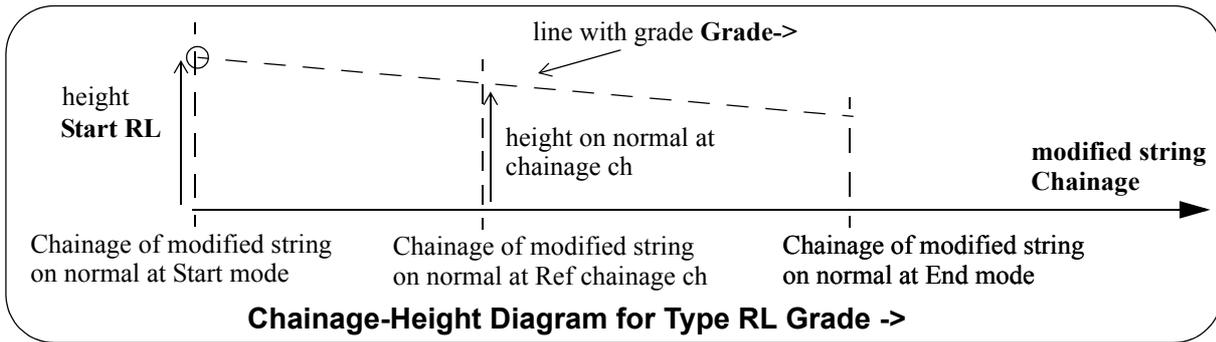
Hence for the modifiers of *Type Pos -> Pos*, the calculations at chainage ch are the same as the modifiers "Width/Height/Xfall to RL" with the RL being the interpolated RL at chainage ch.

**24.12.5.8.1.8 if RL Grade ->: start with a given Start RL and continue on the line at grade Grade->**



The height at the vertex on the normal at **Start mode** is the **Start RL**.

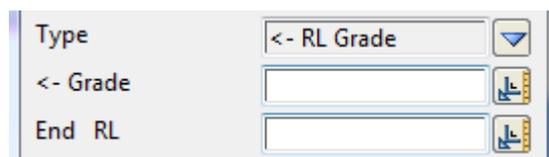
For each normal between the *Start mode* and the *End Mode*, the height of the vertex is on the line with grade **Grade->** with respect to the modified string, and going through the point (normal at Start mode, Start RL).



Hence for the modifiers of *Type RL Grade ->*, the calculations at chainage ch are the same as the modifiers "Width/Height/Slope to RL" with the RL being that calculated at ch using the **Start RL** and the **Grade->**.

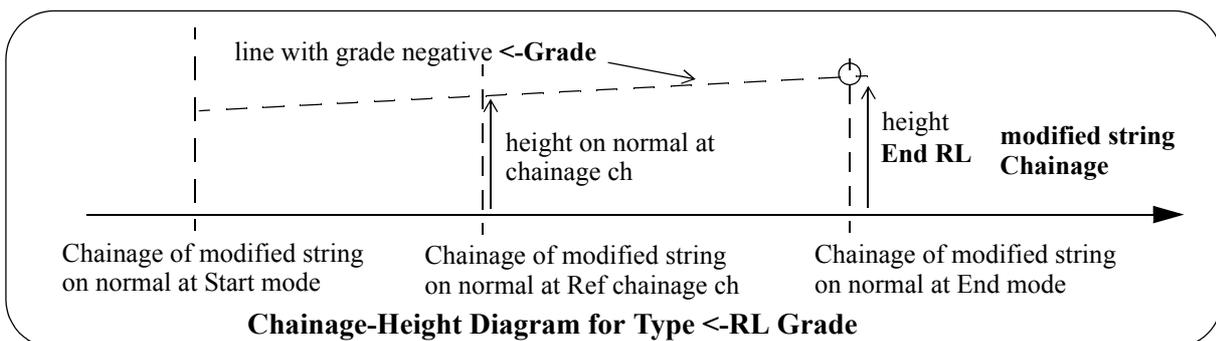
**Important Note:** Grade is calculated using **increasing Reference chainages**.

**24.12.5.8.1.9 if <- RL Grade: all points are on a line with given grade and going through End RL**



The height on the vertex at **End mode** is **End RL**.

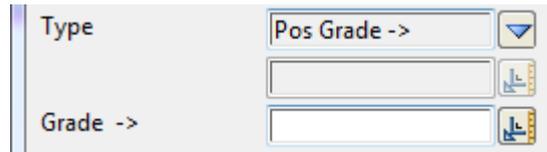
For each normal between the *Start mode* and the *End Mode*, the height of the vertex is on the line with grade of negative **<-Grade** with respect to the modified string chainage, and going through the point (normal at End mode, End RL).



Hence for the modifiers for *Type <-RL Grade*, the calculations at chainage ch are the same as the modifiers "Width/Height/Xfall to RL" with the RL being that calculated at ch using the **End RL** and the **Grade**.

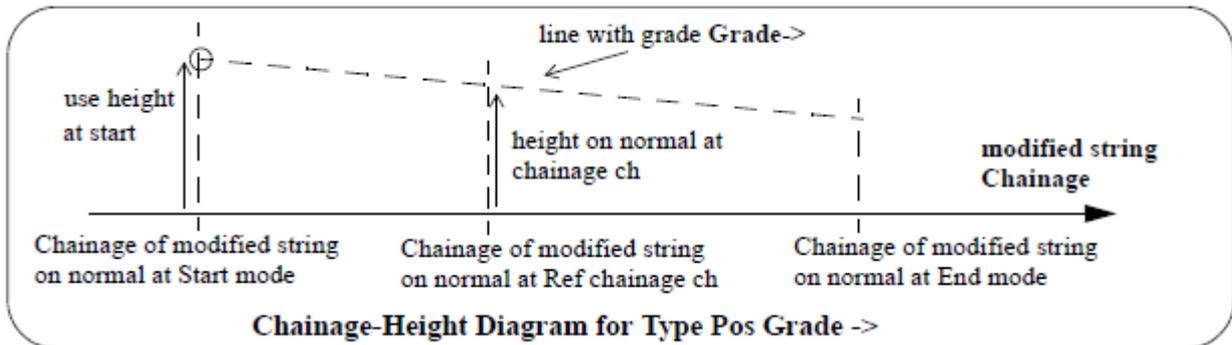
**Important Note:** <-Grade is calculated using **decreasing Alignment chainages**. So in a diagram with increasing chainage, the grade is negative **<-Grade**.

**24.12.5.8.1.10 if Pos Grade ->: start with the calculated height and continue at a given grade**



First let **Start RL** be the actual height calculated at the vertex on the normal at **Start mode** when all the modifiers before this modifier are applied.

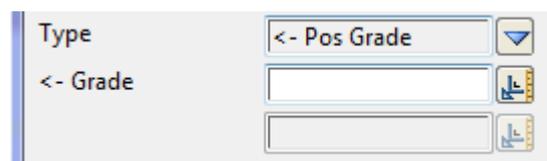
Then the height of each vertex on each normal between the *Start mode* and the *End Mode* is on the line with grade **Grade->** with respect to the modified string chainage, and going through the point (normal at Start mode, Start RL).



Hence for the modifiers of *Type Pos Grade->*, the calculations at chainage ch are the same as for the modifiers "Width/Height/Xfall to RL" with the RL being calculated at ch using the **Start RL** and the **Grade**.

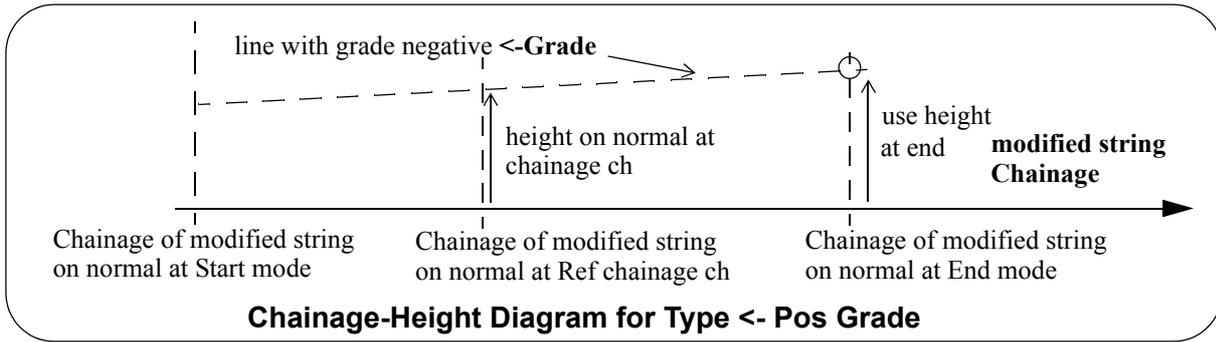
**Important Note:** Grade is calculated using **increasing Alignment chainages**.

**24.12.5.8.1.11 if <- Pos Grade: all points are on a line of given grade and going through the end point**



First let **End RL** be the actual height on the vertex on the normal at **End mode** when all the modifiers before this modifier are applied.

Then the height of each vertex on the normals between the *Start mode* and the *End Mode* is on the line with grade of negative **<-Grade** with respect to the modified string chainage, and going through the point (normal at End mode, End RL).



Hence for the modifiers of *Type <- Pos Grade*, the calculations at chainage *ch* are the same as the modifiers "Width/Height/Xfall to RL" with the RL being calculated at *ch* using the **End RL** and the **<-Grade**.

**Important Note:** <-Grade is calculated using **decreasing Alignment chainages** and the **heights at the points** along the string being modified. So in a diagram with increasing chainage, the grade is negative **<-Grade**.

#### 24.12.5.8.1.12 For Type Pos-> Pos: Tangential Compound Parabolas

First let **Start RL** be the actual height calculated at the vertex of the string at the normal to **Start mode** when all the modifiers before this modifier are applied.

Let **End RL** be the actual height calculated at the vertex of the string at the normal to **End mode** when all the modifiers before this modifier are applied.

Then for each vertex on the normals between **Start mode** and **End Mode**, the height of the vertex is on the curve of tangential compound parabolas (in chainage-height space) between the *Start RL* and the *End RL*.

#### 24.12.5.8.1.13 For Type Pos-> Pos: Tangential Cubic

First let **Start RL** be the actual height calculated at the vertex of the string at the normal to **Start mode** when all the modifiers before this modifier are applied.

Let **End RL** be the actual height calculated at the vertex of the string at the normal to **End mode** when all the modifiers before this modifier are applied.

Then for each vertex on the normals between **Start mode** and **End Mode**, the height of the vertex is on the curve of tangential cubic (in chainage-height space) between the *Start RL* and the *End RL*.

### 24.12.5.9 Left/Right Modifiers Cut =>by 2 strings

The **Cut =>by 2 strings** option calculates the width, height and xfall or slope from two given strings.

A major difference to V10 is that in V11 it doesn't matter which of the two values from width, height and slope were used to defined the fixed link, the **by 2 strings** option allows for any combination to specify how to **modify** one value to take it from between two strings whilst holding another one to what it is defined as by the link.



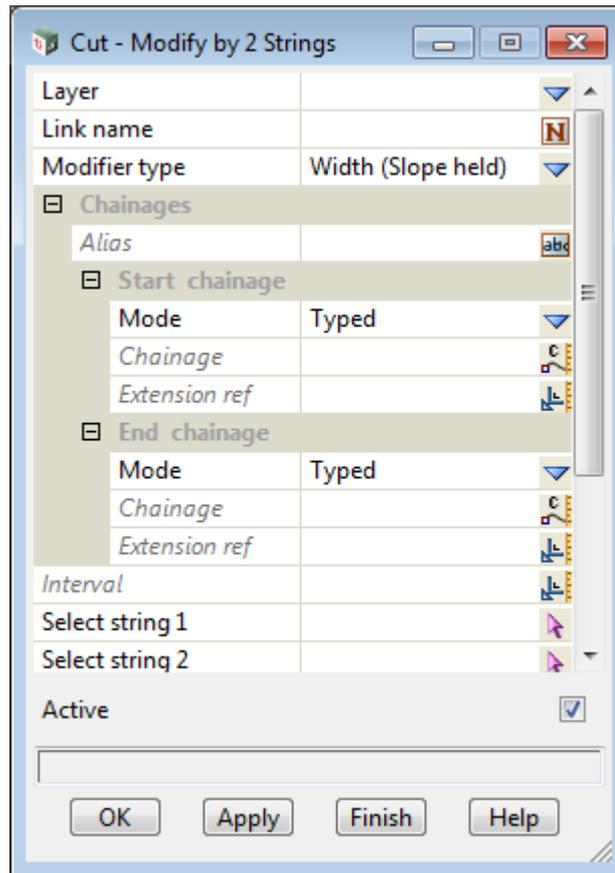
So the **by 2 strings** option replaces the V10 **Cut** options

**Cut =>to 2 strings =>Width to 2 strings**

**Cut =>to 2 strings =>Height to 2 strings**

**Cut =>to 2 strings =>Slope to 2 strings**

Selecting the **by 2 strings** brings up the **Cut - Modify by 2 Strings** panel.



The fields and buttons used in this panel have the following functions.

Field	Description	Type	Defaults	Pop-Up
<b>Layer</b>	<i>name of the layer that the link to be modified is in.</i>	layer box	Design	available layers
<b>Link name</b>	<i>the name of the link in the Layer to modify the width/height/xfall/slope using the method given in <b>Modifier type</b>, between the chainages given by <b>Start mode</b> and <b>End mode</b>. This is the current link.</i>	name box		names.4d pop-up
<b>Modifier type</b>		choice box		

- modify Width and take Height from current link
- modify Width and take Slope from current link
- modify Height and take Width from current link
- modify Height and take Slope from current link
- modify Slope and take Width from current link
- modify Slope and take Height from current link



*A link can be modified in any way regardless of how the link was defined in terms of width, height and xfall or slope. See [24.12.1 Calculating Width, Height, Xfall or Slope from Original Definitions](#).*

*The **Modify** width/height/slope is the part of the selected link that is modified from its current value by*

the method given by **Modifier type**.

The **Hold** width/height/slope is the part of the selected link that is taken from the current value for the link.

For **Modify Width**, see [24.12.5.9.1 Cut =>by 2 strings Modifier Types "Modify Width. Hold Height/Slope"](#)

For **Modify Height**, see [24.12.5.9.2 Cut =>by 2 strings Modifier Types "Modify Height. Hold Width/Slope"](#)

For **Modify Slope**, see [24.12.5.9.3 Cut => by 2 strings Modifier Types "Modify Slope. Hold Width/Height"](#)

**String 1** string-select

select the first string to use for defining width/height/slope for the link.

**String 2** string-select

select the second string to use for defining width/height/slope for the link.

**Side 1 to search** choice box left side left side, right side, both sides  
side of the hinge string to start searching to find string 1 to use in defining width/height/slope.

**Side 2 to search** choice box left side left side, right side, both sides  
side of the hinge string to start searching to find string 2 to use in defining width/height/slope.

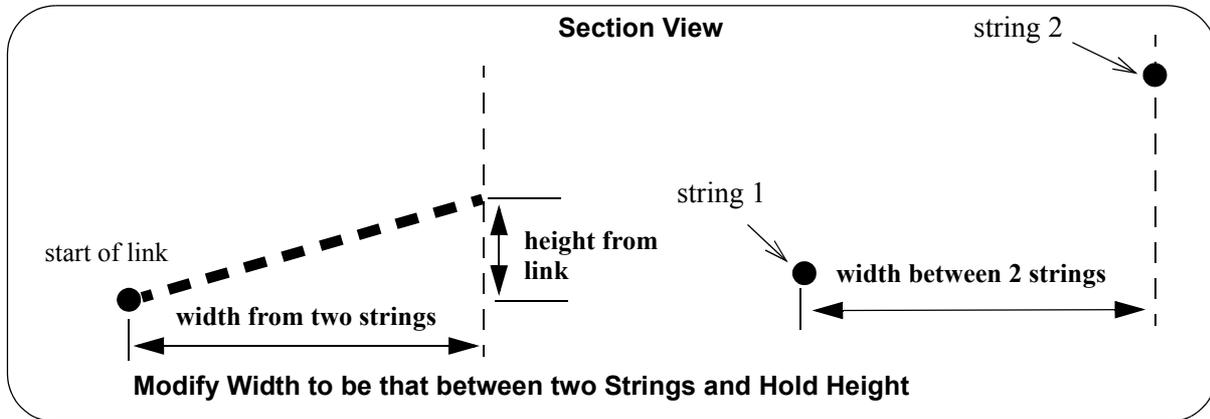
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).

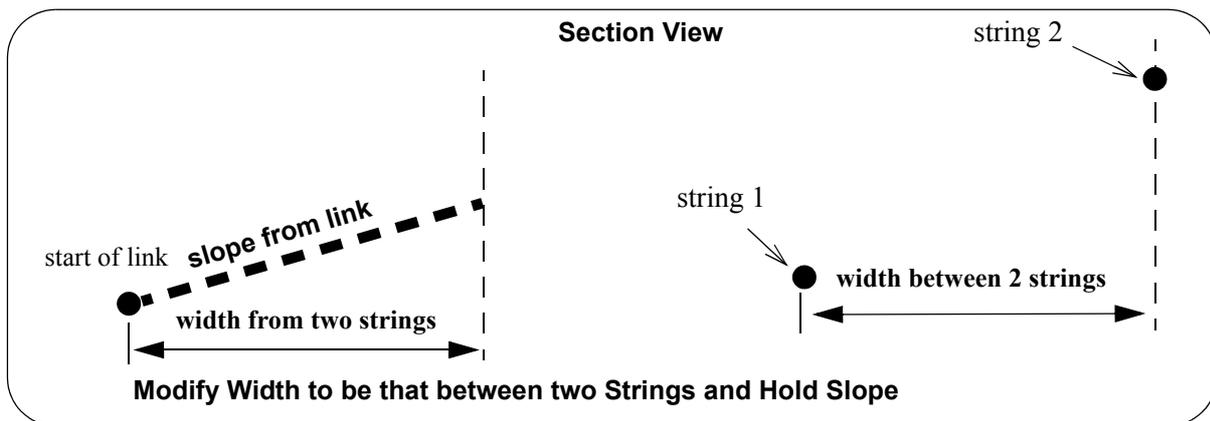
### 24.12.5.9.1 Cut =>by 2 strings Modifier Types "Modify Width, Hold Height/Slope"

For a cut link **Modify Width** sets the **width** of the link to be the width between the two selected strings.

For **Modify Width, Hold Height** the *height* is taken from the link.



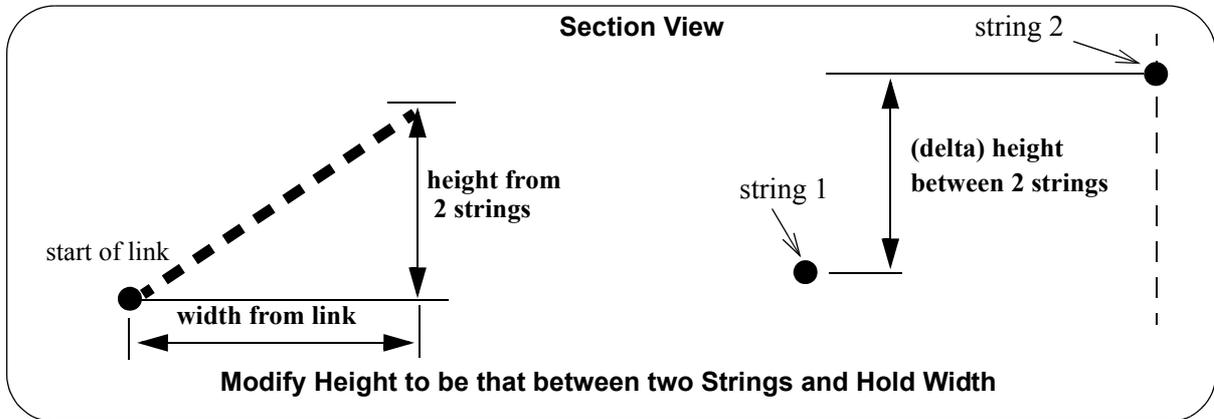
For **Modify Width, Hold Slope** the *slope* is taken from the link.



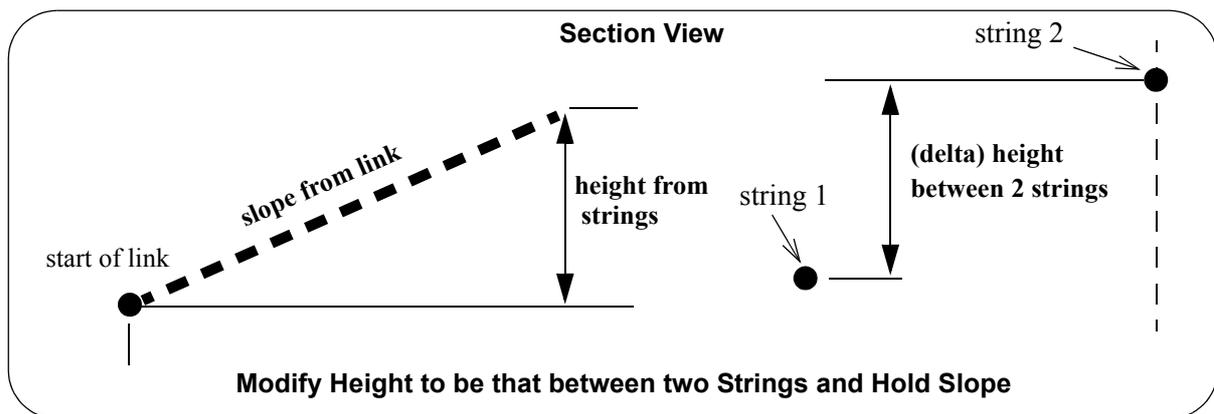
### 24.12.5.9.2 Cut =>by 2 strings Modifier Types "Modify Height, Hold Width/Slope"

For a cut link **Modify Height** sets the **height** of the link to be the height between the two selected strings.

For **Modify Height, Hold Width** the *width* is taken from the link.



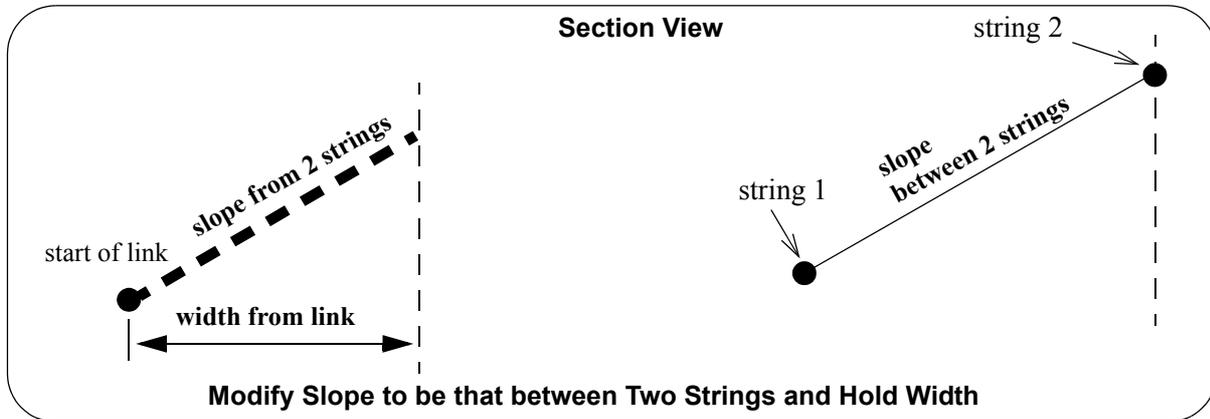
For **Modify Height, Hold Slope** the *slope* is taken from the link.



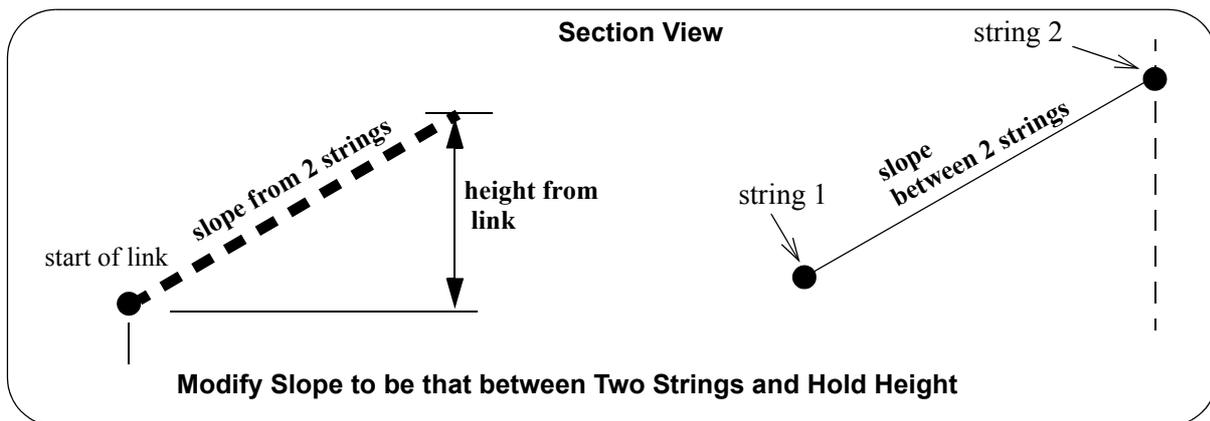
### 24.12.5.9.3 Cut => by 2 strings Modifier Types "Modify Slope, Hold Width/Height"

For a cut link **Modify Slope** sets the **slope** of the link to be the slope between the two selected strings.

For **Modify Slope, Hold Width** the *width* is taken from the link.



For **Modify Slope, Hold Height** the *height* is taken from the link.



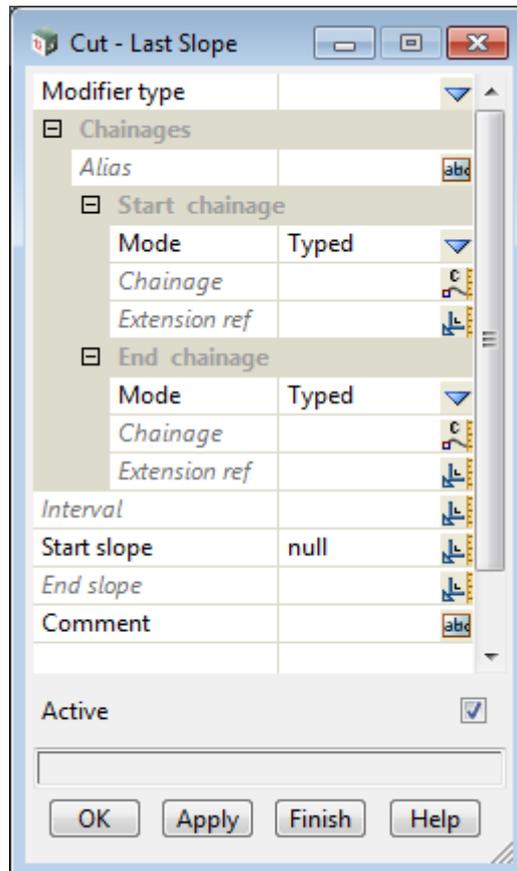
Note that the slope between the two strings may go up or it may go down.

### 24.12.5.10 Left/Right Modifiers Cut =>Last slope

This is new in V11.

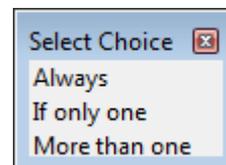
The **Cut =>last slope** option modifies the last slope in the Cut table and allows you to differentiate between the cases when only one or more than one link was created by the Cut table.

Selecting the **Last slope** brings up the **Cut - Last Slope** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Modifier type</b>	choice box		



*if **Always**, this modifier is used when one or more links are created by the Cut table. This modifier applies to the last cut link created.*

*If **If only one**, this modifier is only used if exactly one link is created by the Cut table. This modifier applies to this one cut link created.*

*If **More than one**, this modifier is only used if more than one link is created by the Cut table. This modifier applies to the last cut link created.*

<b>Start slope</b>	real box
--------------------	----------

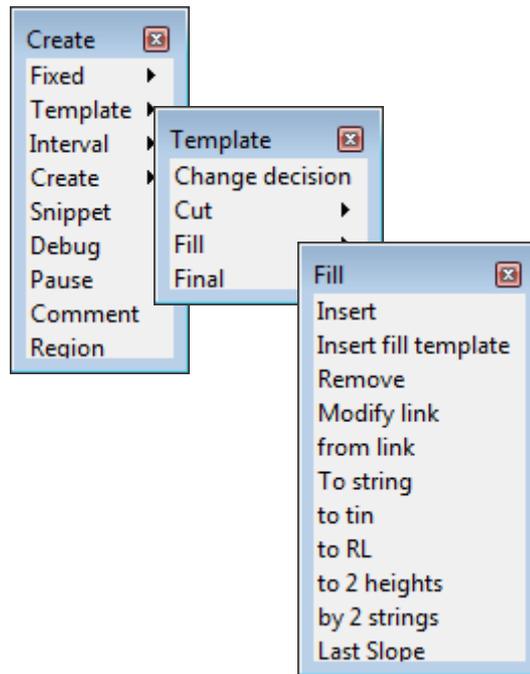
*the slope to use at the Start chainage. The slope is linearly interpolated with respect to reference chainage between the **Start slope** and the **End slope**.*

**End slope**                      real box  
*the slope to use at the End chainage.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

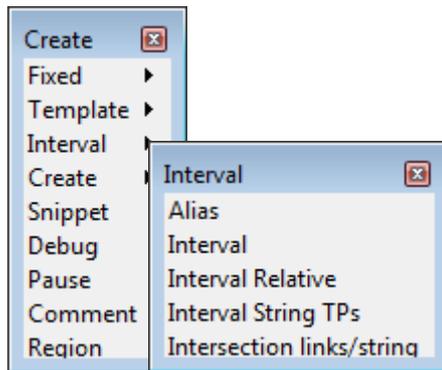
*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#) .*

## 24.12.6 Left/Right Modifiers => Create =>Template =>Fill



*This section has not yet been documented but it is virtually identical to the Create =>Cut section except the links in the diagrams go down rather than up.*

## 24.12.7 Left/Right Modifiers => Create =>Interval



See

[24.12.7.1 Left/Right Modifiers => Interval =>Alias](#)

[24.12.7.2 Left/Right Modifiers => Interval =>Interval](#)

[24.12.7.3 Left/Right Modifiers => Interval =>Interval Relative](#)

[24.12.7.4 Left/Right Modifiers => Interval =>Interval String TPs](#)

### 24.12.7.1 Left/Right Modifiers => Interval => Alias

The **Alias** option creates a row in the grid with a given **Alias** name.

The Alias is then available to be used by *Smart Chainages*.

For the new information on **Smart Chainage**, see [9. Smart Chainages](#).

Selecting **Alias** brings up the panel **Chainage Alias**

**Alias** text box

*a text name that can be used to refer to the row in the grid for this command.*

See [24.8 Alias for Left and Right Modifiers](#).

### 24.12.7.2 Left/Right Modifiers => Interval =>Interval

In V10 this option was **Fixed =>Interval**.

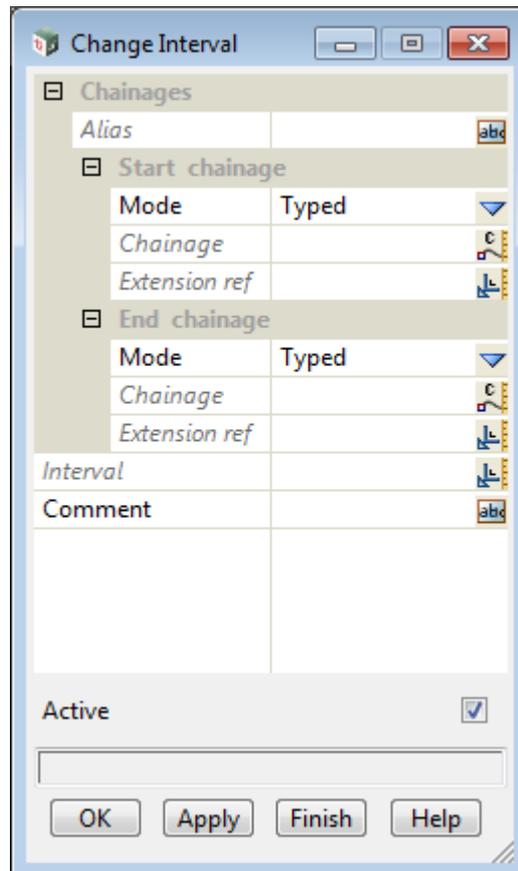
The **Interval** option changes the interval between the MTF sections over the given chainage range.

The sections are at the chainages that are integer multiples of the interval, and less than or equal to the start chainage and less than the end chainage. That is, the chainage values are referenced about zero and inside the start/end range.

For example, if the interval is 10 and the start chainage is -13 and the end chainage is 24 then there will be sections as

-10, 0, 10 and 20

The default value used for Interval is the **Separation distance** in the **Apply MTF** panel.



**Alias** text box

*if **not blank**, a text name that can be used to refer to the row in the grid for this command.*

See [24.8 Alias for Left and Right Modifiers](#)

### 24.12.7.3 Left/Right Modifiers => Interval =>Interval Relative

This is a new option.

The **Interval Relative** option changes the interval between the MTF sections over the given chainage range but instead of being referenced about zero, the chainages are reference about the start chainage.

That is, the sections are at the chainages that are the start chainage plus integer multiples of the interval, and less than or equal to the start chainage and less than the end chainage. That is, the chainage values are referenced about the start chainage and inside the start/end range.

For example, if the interval is 10 and the start chainage is -13 and the end chainage is 24 then there will be sections as

-13, -3, 7 and 17

**Alias** text box

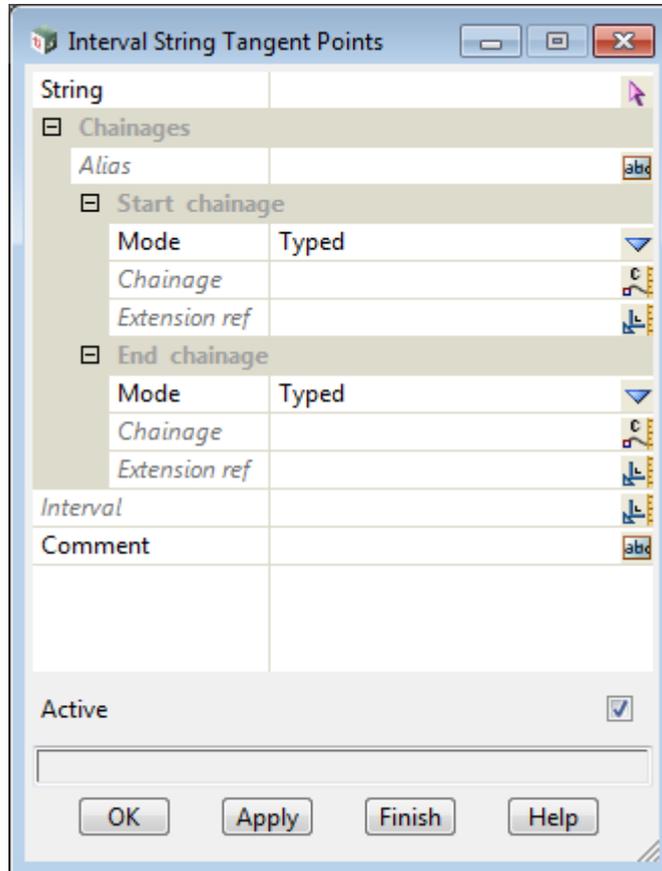
if **not blank**, a text name that can be used to refer to the row in the grid for this command.

See [24.8 Alias for Left and Right Modifiers](#).

### 24.12.7.4 Left/Right Modifiers => Interval =>Interval String TPs

This is a new option.

The **Interval String TPs** option drops the tangents points of a selected string onto the reference string and includes the dropped points as extra chainages where sections are generated.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>String</b>	string select		

*the string to select whose tangent points are dropped onto the reference string and if any of the dropped points are in the start and end chainage range then sections are generated at those dropped chainages.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

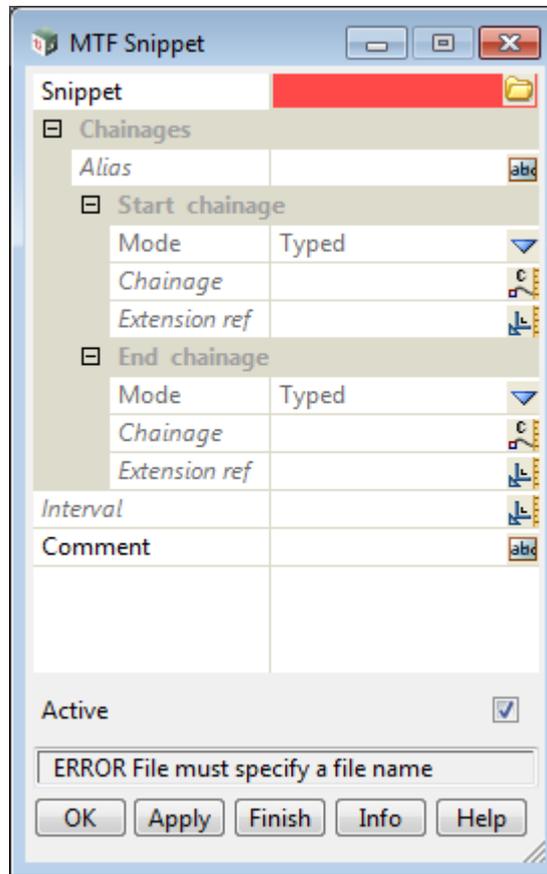
*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 24.12.8 Left/Right Modifiers => Create => Snippet

Snippets have been considerably enhanced in V11.

For more information on the additions to Snippets, see [24.13 Defining and Using Snippets](#).

Clicking on **Snippet** brings up the **MTF Snippet** panel which is used to insert snippets.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Snippet</b>	snippet box		*.mtfsnippet, *.mtfsnippetc files

*snippet to run.*

*For more information on the inserting and use of snippets, see [24.13 Defining and Using Snippets](#).*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 24.12.9 Left/Right Modifiers - Creating Strings, Shapes and Tins

See

[24.12.9.1 Left/Right Modifiers => Create =>Create =>Shapes](#)

[24.12.9.2 Left/Right Modifiers => Create =>Create =>Strings](#)

[24.12.9.3 Left/Right Modifiers => Create =>Create =>Tins](#)

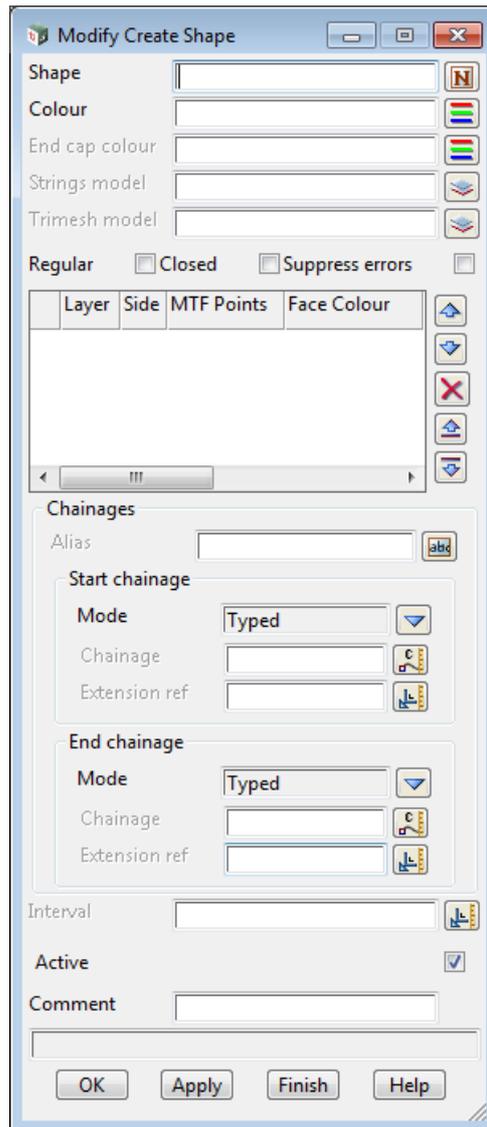
### 24.12.9.1 Left/Right Modifiers => Create =>Create =>Shapes

This is new in V11. Yes the panel is very long.

The **Shapes** option creates a shape from nominated MTF Points, and optionally strings and trimeshes.

For information on MTF Points, Strings, Shapes and Trimeshes, see [24.1 MTF Links, Points, Sections, Strings and Trimeshes](#).

Selecting **Shape** brings up the panel **Modify Create Shape**



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Shape</b> <i>name for the new shape</i>	text box		
<b>Colour</b>	colour box		available colours

colour of the shape.

**Strings model**                      model box                      available models  
*if **not blank**, strings are created for the MTF point listed in the Shape grid and added to this model. The strings have the same name and colour as the MTF points.*

*If **blank**, strings are not created.*

**Trimesh model**                      model box                      available models  
*if **not blank**, a trimesh is created for the shape and added to this model.*

*If **blank**, trimeshes are not created.*

**Regular**                                  tick box  
*if **ticked** the shape will only be formed when all of the nominated strings exist. E.g. it will form multiple shapes if strings come and go along the length of the apply.*

*If **not ticked**,.*

**Closed**                                  tick box  
*if **ticked**, the shape is closed off. That is, the last MTF point in the list is joined to the first MTF point.*  
*If **ticked**, the shape is not closed off.*

**Shape Grid**

**Layer**                                  layer box                      available Layers  
*layer to take the MTF point from.*

**Side**                                  choice box                      Left, Right  
*side of the template to take the MTF point from.*

**Links L J G ??**                      text box  
*the name of the MTF point from the layer **Layer** and side **Side**.*

*The shape profile is constructed from the shape grid defines by joining each MTF point to the next MTF point in the list. See [24.1 MTF Links, Points, Sections, Strings and Trimeshes](#)*

*If **Closed** is ticked, the last MTF point in the list is joined to the first MTF point.*

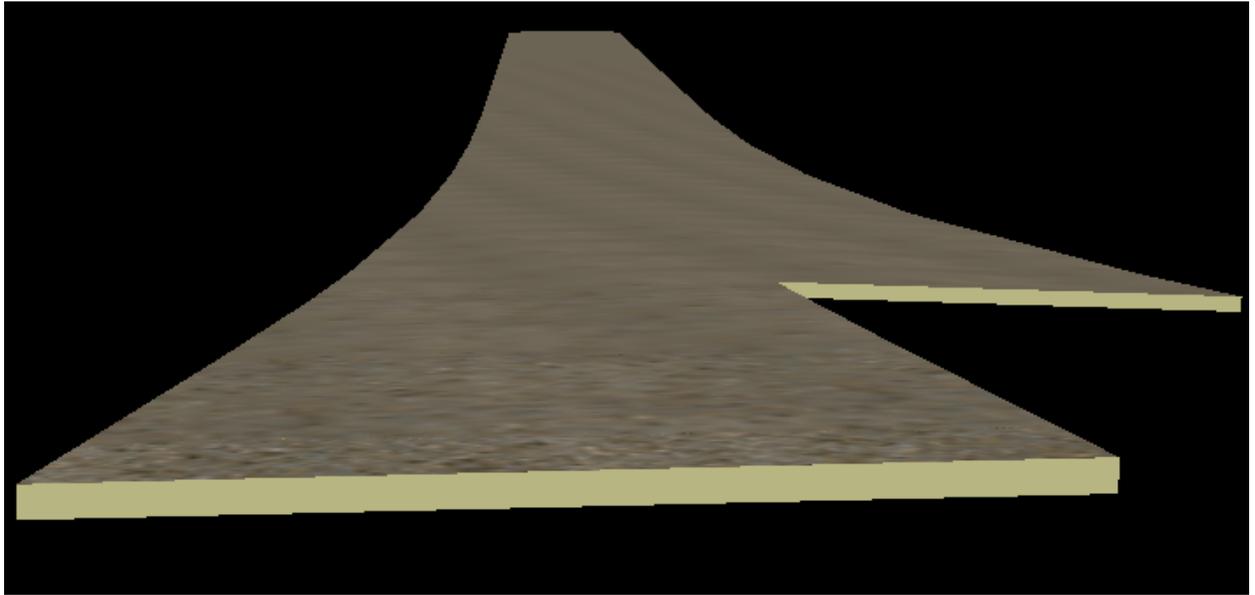
**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

For example, the list of MTF points

Closed <input checked="" type="checkbox"/>			
	Layer	Side	Links
1	Desigr	Left	HS
2	Desigr	Left	LKL
3	AC	Left	LKL
4	AC	Left	HS

creates a shape at each chainage by joining the four MTF points in the order down the list, and then closes the shape by joining the fourth MTF point to the first MTF point.



## 24.12.9.2 Left/Right Modifiers => Create =>Create =>Strings

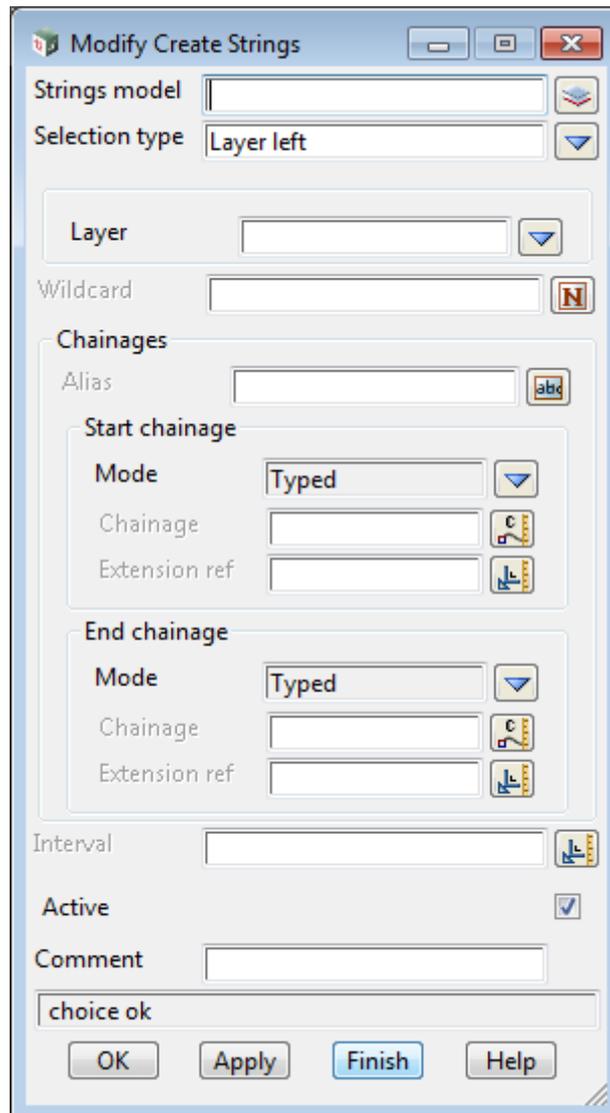
This is new in V11.

The **Strings** option creates longitudinal strings for nominated MTF points and places them in a given model.

One application of the **Strings** option is to create a model of strings for a given Layer instead of the **Boxing Layers** in the **Models** tab of the **Apply MTF Function**.

**Note:** Strings and sections are automatically created for all the MTF points in the default Layer Design.

Selecting **Strings** brings up the panel **Modify Create Strings**

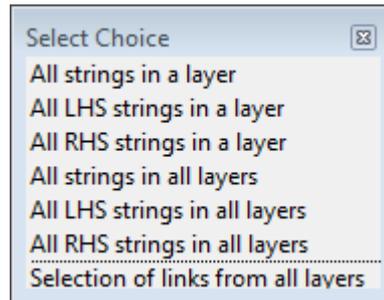


The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Strings model</b>	model box		available models

*the strings that created for the nominated points are placed in this model.*

**Selection type** choice box



*Selection type* specifies where to get all the names of points created at any section in the Start and End chainage range.

For any **Selection type** other than **Selection of links from all layers**, the points then have to match the **Wildcard** and then strings are created with the name and colour of the point, and placed in the **Strings** model.

If **All points in a layer**, all points in the layer given in **Layer**.

If **All LHS points in a layer**, all points in the layer given in **Layer** that are on the left hand side.

If **All RHS points in a layer**, all points in the layer given in **Layer** that are on the right hand side.

If **All points in all layers**, all points in the all layers.

If **All LHS points in all layers**, all points in the all layers that are on the left hand side.

If **All RHS points in all layers**, all points in the all layers that are on the right hand side.

**Layer** layer box available Layers

used when **Selection type** is All points in a layer, All LHS points in a layer, or All RHS points in a layer.

**Wildcard** text box

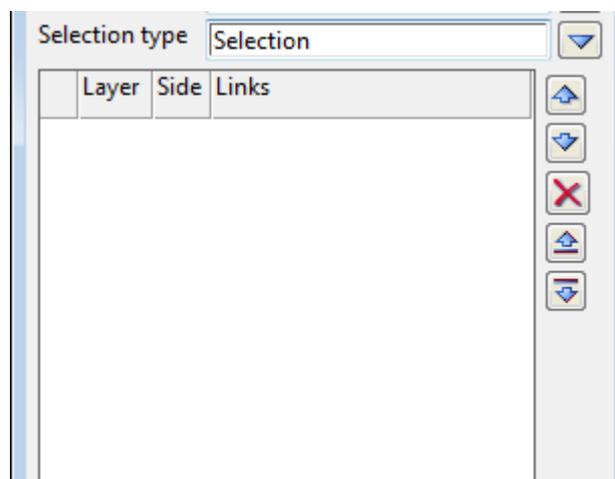
used for all **Selection type's** except Selection of points for all layers.

text including wild cards \* and wild characters ?.

If **blank**, all points from **Selection type** have strings created for them

If **not blank**, any names of points from the **Selection type** that **match** the **Wildcard** have strings created for them with the same name and colour as the point.

If **Selection of points for all layers**, then a grid is used to nominate the points to be selected. The points do not have to be from the same layer.



**Selection Grid**

**Layer**                    layer box                    available Layers  
*layer to take the point from.*

**Side**                    choice box                    Left, Right  
*side of the template to use to take the point.*

**Points**                    text box  
*the name of the point from the layer **Layer** and side **Side**.*

*For any of the points in the table, strings are created and placed in the **Strings model**.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

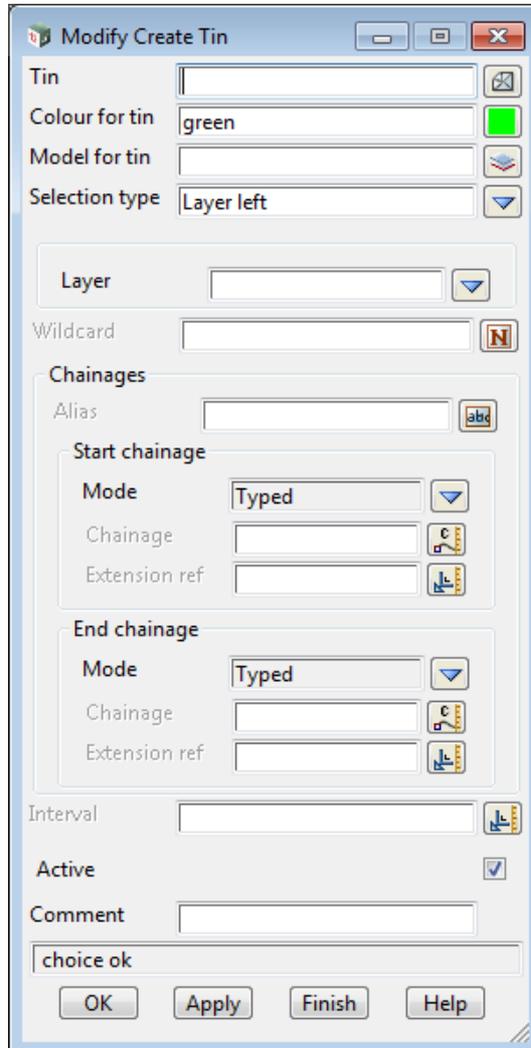
*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### 24.12.9.3 Left/Right Modifiers => Create =>Create =>Tins

This is new in V11.

The **Tins** option creates a tin from selected MTF strings.

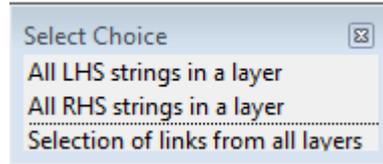
Selecting **Tins** brings up the panel **Modify Create Tin**



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Tin</b> <i>the name for the created tin.</i>	tin box		available tins
<b>Colour for tin</b> <i>the colour of the created tin.</i>	colour box		available colours
<b>Model for tin</b> <i>the created tin is added to this model.</i>	model box		available models

**Selection type** choice box



***Selection type** specifies where to get all the MTF points in sections in the Start and End chainage range, and create the strings that are used in the triangulation.*

*For any **Selection type** other than **Selection of links from all layers**, the points have to match the **Wildcard**. Then strings are created and used in the triangulation.*

*If **All LHS strings in a layer**, then all points in the layer given in **Layer** that are on the left hand side and satisfy **Wildcard** are used to create strings, and the strings are used in the triangulation.*

*If **All RHS strings in a layer**, then all points in the layer given in **Layer** that are on the right hand side and satisfy **Wildcard** are used to create strings, and the strings are used in the triangulation.*

**Layer** layer box available Layers

*used for all **Selection type's** except Selection of points for all layers.*

*The Layer to take MTF points from.*

**Wildcard** text box

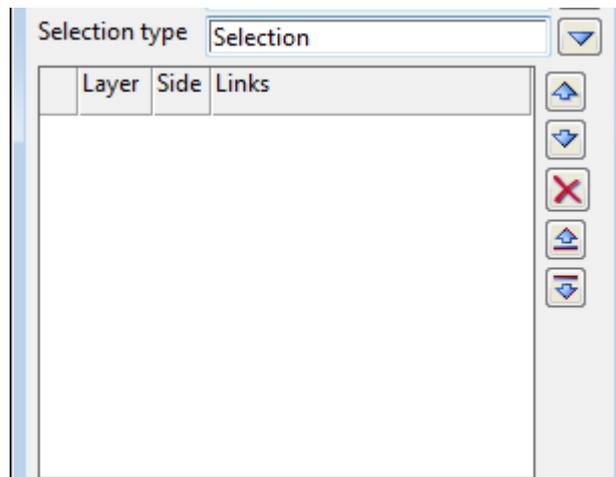
*used for all **Selection type's** except Selection of points for all layers.*

*Text including wild cards \* and wild characters ?.*

*If **blank**, all points from **Selection type** have strings created for them, and the strings are used in the triangulation.*

*If **not blank**, any names of points from the **Selection type** that **match** the **Wildcard** have strings created for them and the strings are used in the triangulation.*

*If **Selection of points for all layers**, then a grid is used to nominate the points to be selected. The points do no have to be from the same layer.*



**Selection Grid**

**Layer** layer box available Layers

*layer to take the point from.*

**Side** choice box Left, Right

*side of the template to use to take the point.*

**Points** text box

*the name of the point from the layer **Layer** and side **Side**.*

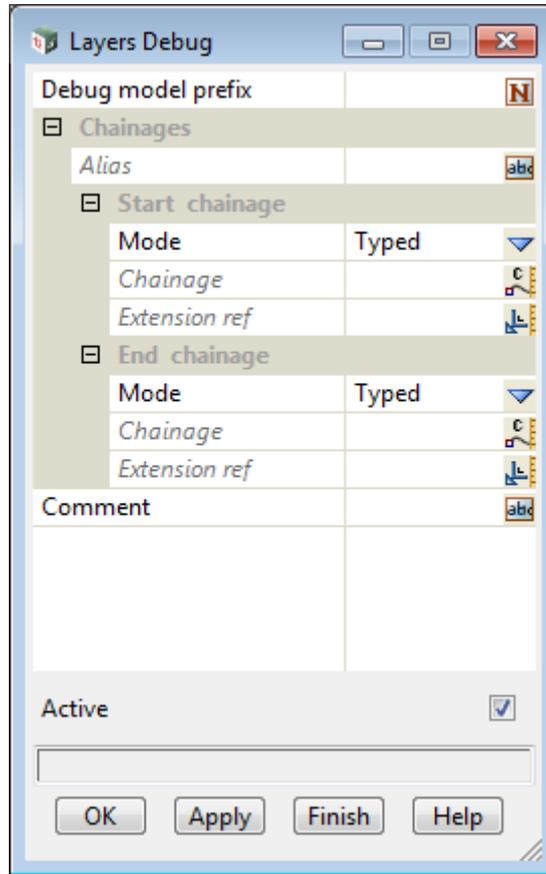
*For any of the points in the table, strings are created and placed in the **Strings model**.*

**Alias, Start/End Change, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 24.12.10 Left/Right Modifiers => Create => Debug

Clicking on **Debug** brings up the **Layers Debug** panel which is used to create debug information when you are having problems with your MTF, especially with Snippets.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Debug model prefix</b>	text box		

*debug models are created with the name of this prefix followed by the chainage.*

### **Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

### **What the Panel Does**

*For each chainage, a model is created with the name **Debug model prefix** followed by the chainage value. In the model all the link in all the layers are drawn but with the offset as the x-value and height as the y-value so that the model can be looked at on a **Plan View**.*

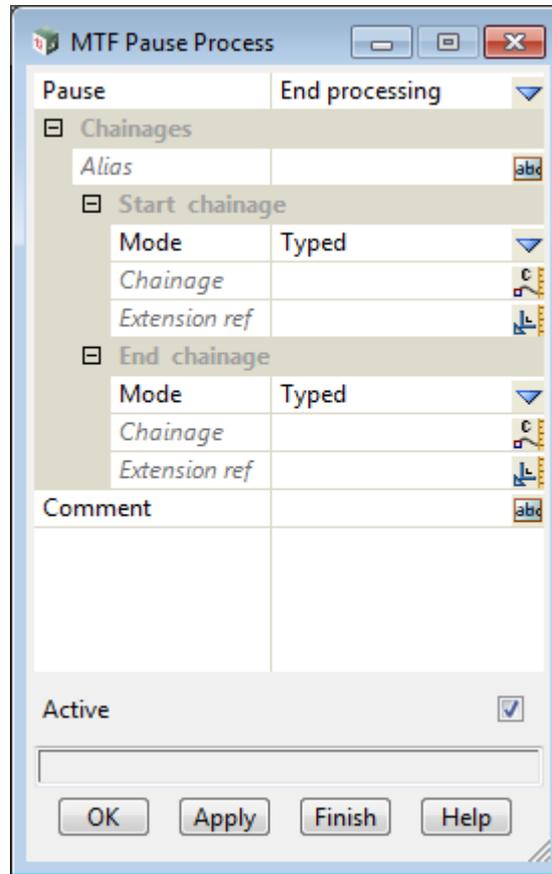
*A report file is also created at the start chainage and it contains information debugging such as the snippets used and the values of the parameters in the snippets.*

*The Report has the name **Debug model prefix** followed by the chainage value with the decimal point replaced by an underscore, with **12D\_APPLY\_MTF\_DEBUG\_DUMP** as the file extension.*

*For example, **Lee Road debug 0\_000.12D\_APPLY\_MANY\_DEBUG\_DUMP***

## 24.12.11 Left/Right Modifiers => Create =>Pause

Clicking on **Pause** brings up the **MTF Pause Process** panel which is used to enable/disable the processing of all the MTF commands in the given chainage range.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Pause</b>	choice box	Stop processing	Start processing, Stop processing

*if **End processing**, then for a given chainage **ch** in the chainage range, all commands after this command will **NOT** be processed until a **MTF Pause Process** panel with **Process** set to **Start processing** (and whose chainage range includes this chainage **ch**) is met.*

*If **Start processing**, then for a given chainage **ch** in the chainage range, all commands after this command will be processed until a **MTF Pause Process** panel with **Process** set to **End processing** (and whose chainage range includes this chainage **ch**), is met.*

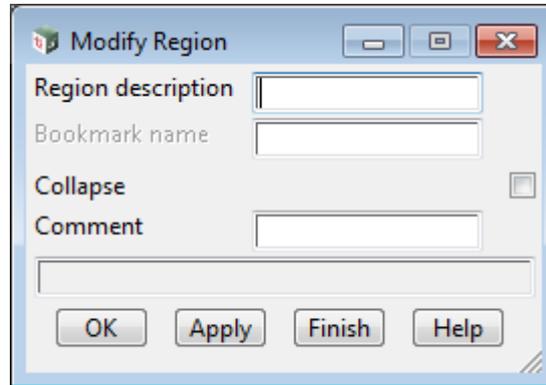
**Alias, Start/End Chainage, Interval, Comment, Extra start, Extra End, Active, OK, Apply**

*For information on these panel fields, see [24.12.2 Common Fields on Modifier Panels](#).*

## 24.12.12 Left/Right Modifiers => Create =>Region

A **Region** in a grid is a special command that has the property that a Region can be **collapsed**. And *collapsing* a Region means that all the rows after the Region command until the next Region command, are hidden in the grid.

The row of the Region command in the grid is coloured light blue.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Region description</b>	text box		
---------------------------	----------	--	--

*name for the Region.*

<b>Bookmark name</b>	text box		
----------------------	----------	--	--

*if **not blank**, this is a secondary name, often a shorter name, that is used in the **Region** pop up rather than using the **Region description**.*

*If **blank**, then when the grid is saved, the **Bookmark name** is set to be the same as the **Region description**.*

<b>Collapse</b>	tick box		
-----------------	----------	--	--

*if **ticked**, all the commands in the grid until the next **Region** command are collapsed into this Region command.*

*If **not ticked**, the commands in the grid until the next **Region** command are not collapsed.*

<b>Comment</b>	text box		
----------------	----------	--	--

*comment to add to the Comment column. In the text file, the comment will be preceded by //.*

<b>OK</b>	button		
-----------	--------	--	--

***OK** stores the values in the fields and removes the panel BUT no **recalc** is done.*

<b>Apply</b>	button		
--------------	--------	--	--

***Apply** stores the values and leaves the panel on the screen.*

*If the **MTF** is being used in an **Apply MTF** and **Auto recalc** is ticked in the MTF, then whenever the **Apply** button is clicked, a **recalc** of the associated **Apply MTF** for the MTF is done.*

## 24.13 Defining and Using Snippets

The sections on snippets are:

- [24.13.1 What are MTF Snippets?](#)
- [24.13.2 Creating a Snippet](#)
- [24.13.3 An Example of a Snippet](#)
- [24.13.4 How To Select a Snippet?](#)
- [24.13.5 Comments in Snippets](#)
- [24.13.6 Start and End Modes for a Snippet](#)
- [24.13.7 Snippet Parameters](#)
- [24.13.8 Automatic Parameters in Snippets](#)
- [24.13.9 Arithmetic in Snippets](#)
- [24.13.10 Trig and Maths Function Capabilities in Snippets](#)
- [24.13.11.1.1 #define in Snippets - Do Not Use](#)
- [24.13.11 Snippet Directives](#)
- [24.13.12 Snippet Variables](#)
- [24.13.13 Order of snippet processing](#)
- [24.13.14 Debugging Snippets](#)
- [24.13.15 Compiling Snippets](#)
- [24.13.16 Tips and Tricks](#)
- [24.13.17 Major Warning - You Will be Caught by This](#)

## 24.13.1 What are MTF Snippets?

A snippet is a collection of many single MTF modifiers that can be inserted as a group into an MTF file.

However unlike an MTF file or an insert in an MTF, values to be used inside the snippet can be passed down to the snippet via parameters whose values are specified each time the snippet is placed.

So, snippets enable things such as driveway laybacks, kerb transitions, earthworks flares etc. to be a single entry in a MTF file thus reducing the size of the MTF file and enhancing it's readability.

Setting up a snippet is the same complexity as creating MTF items but once a snippet is created, it can be reused over and over again.

Snippets can also be used to hide complex modifiers from novice users.

Go to the next section [24.13.2 Creating a Snippet](#) or back to [24.13 Defining and Using Snippets](#).

## 24.13.2 Creating a Snippet

There is currently no snippet Create/Editor in **12d Model** so for the moment snippets must be manually created as a text file. Snippet text files have the extension **MTFSNIPPET**.

Note that this is case insensitive. That is, each letter can be in upper or lower case.

Luckily the format for each snippet command is almost identical to the normal modifier commands as written out to the `left_side_modifier` or `right_side_modifier` sections of the MTF file when viewed as a text file. The snippet commands differ slightly in that they can contain variables and parameters that can be substituted when the snippet is run.

So the easiest way to create a snippet file is to create a normal MTF inside **12d Model** with the modifiers to be included and used in the snippet. Once these modifier commands are in the MTF, they can be copied directly from the Modifiers grid in a snippet compatible format.

To do so, select one or more rows of modifiers in the grid. Then right-click on one of the row header to bring up the modifiers grid context menu and select **Copy (snippet paste)**. This will copy the selected commands to the Windows clipboard.

Now that the desired modifier commands have been copied to the clipboard, you can create a new text document in your preferred text editor and paste the commands into the new document. A snippet, at its simplest, can simply be these same modifier commands in a snippet file, however, the real power and flexibility of snippets comes from using the more advanced functionality discussed in the following sections.

It is also worth noting that the snippet commands can be created manually, rather than by copy and paste, by typing the required snippet and modifier commands directly into a text file. Note, however, that the syntax and format of the snippet and modifiers must be correct or you will get errors when the snippet is inserted or run.

How to construct the additional information in the snippet file, and how a snippet works, will be explained in the following sections.

Go to the next section [24.13.3 An Example of a Snippet](#) or back to [24.13 Defining and Using Snippets](#).

## 24.13.3 An Example of a Snippet

The following is an example of a snippet called KERB\_SA\_DW places a layback at a driveway. It will be used as the example in describing the workings of a snippet.

```
// PARAMETER WIDTH REAL "Layback width" 0.6
// PARAMETER DEPTH REAL "Layback depth" -0.04

// drop into DW

insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0 named_position
"MODIFIER_START" 0.5 absolute extra_start extra_end

insert "SAI" "grey" 0.5 $DEPTH unknown named_position "MODIFIER_START" 0 named_position
"MODIFIER_START" 0.5 absolute extra_start extra_end

insert "SAT" "grey" 0.03 0.15 unknown named_position "MODIFIER_START" 0 named_position
"MODIFIER_START" 0.5 absolute extra_start extra_end

insert "SAB" "grey" 0.18 0 unknown named_position "MODIFIER_START" 0 named_position
"MODIFIER_START" 0.5 absolute extra_start extra_end

// push out links

width "SAT" named_position "MODIFIER_START" 0 named_position "MODIFIER_START" 0.5 0.03 $WIDTH
absolute extra_start extra_end

width "SAB" named_position "MODIFIER_START" 0 named_position "MODIFIER_START" 0.5 0.18 0.0 absolute
extra_start extra_end

height "SAT" named_position "MODIFIER_START" 0 named_position "MODIFIER_START" 0.5 0.15 0.2 absolute
extra_start extra_end

height "SAB" named_position "MODIFIER_START" 0 named_position "MODIFIER_START" 0.5 0 0 absolute
extra_start extra_end

// DW to -0.5

insert "SKL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0.5 named_position
"MODIFIER_END" -0.5 absolute extra_start extra_end

insert "SKI" "grey" 0.5 $DEPTH unknown named_position "MODIFIER_START" 0.5 named_position
"MODIFIER_END" -0.5 absolute extra_start extra_end

insert "SKB" "grey" $WIDTH 0.2 unknown named_position "MODIFIER_START" 0.5 named_position
"MODIFIER_END" -0.5 absolute extra_start extra_end

//back out -0.5 to end

insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_END" -0.5 named_position "MODIFIER_END"
0.0 absolute extra_start extra_end

insert "SAI" "grey" 0.5 $DEPTH unknown named_position "MODIFIER_END" -0.5 named_position
"MODIFIER_END" 0.0 absolute extra_start extra_end

insert "SAT" "grey" 0.03 0.15 unknown named_position "MODIFIER_END" -0.5 named_position
"MODIFIER_END" 0.0 absolute extra_start extra_end

insert "SAB" "grey" 0.18 0 unknown named_position "MODIFIER_END" -0.5 named_position "MODIFIER_END"
0.0 absolute extra_start extra_end

// push out links

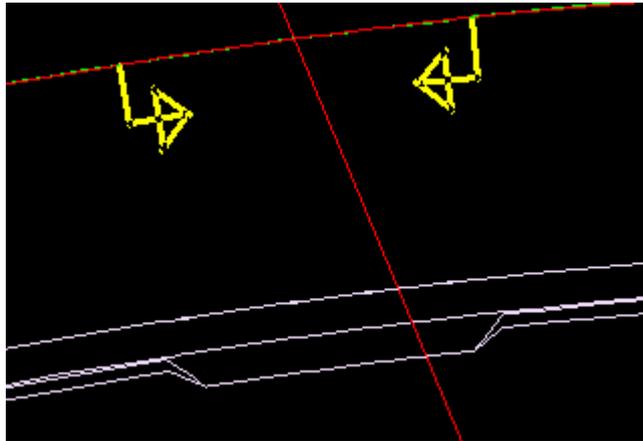
width "SAT" named_position "MODIFIER_END" -0.5 named_position "MODIFIER_END" 0.0 $WIDTH 0.03
```

absolute extra\_start extra\_end

width "SAB" named\_position "MODIFIER\_END" -0.5 named\_position "MODIFIER\_END" 0.0 0.0 0.18 absolute  
extra\_start extra\_end

height "SAT" named\_position "MODIFIER\_END" -0.5 named\_position "MODIFIER\_END" 0.0 0.2 0.15 absolute  
extra\_start extra\_end

height "SAB" named\_position "MODIFIER\_END" -0.5 named\_position "MODIFIER\_END" 0.0 0 0 absolute extra\_start  
extra\_end



**Note:** each line of the snippet is wrapping around because of the limited width of this document. Each line should be like:

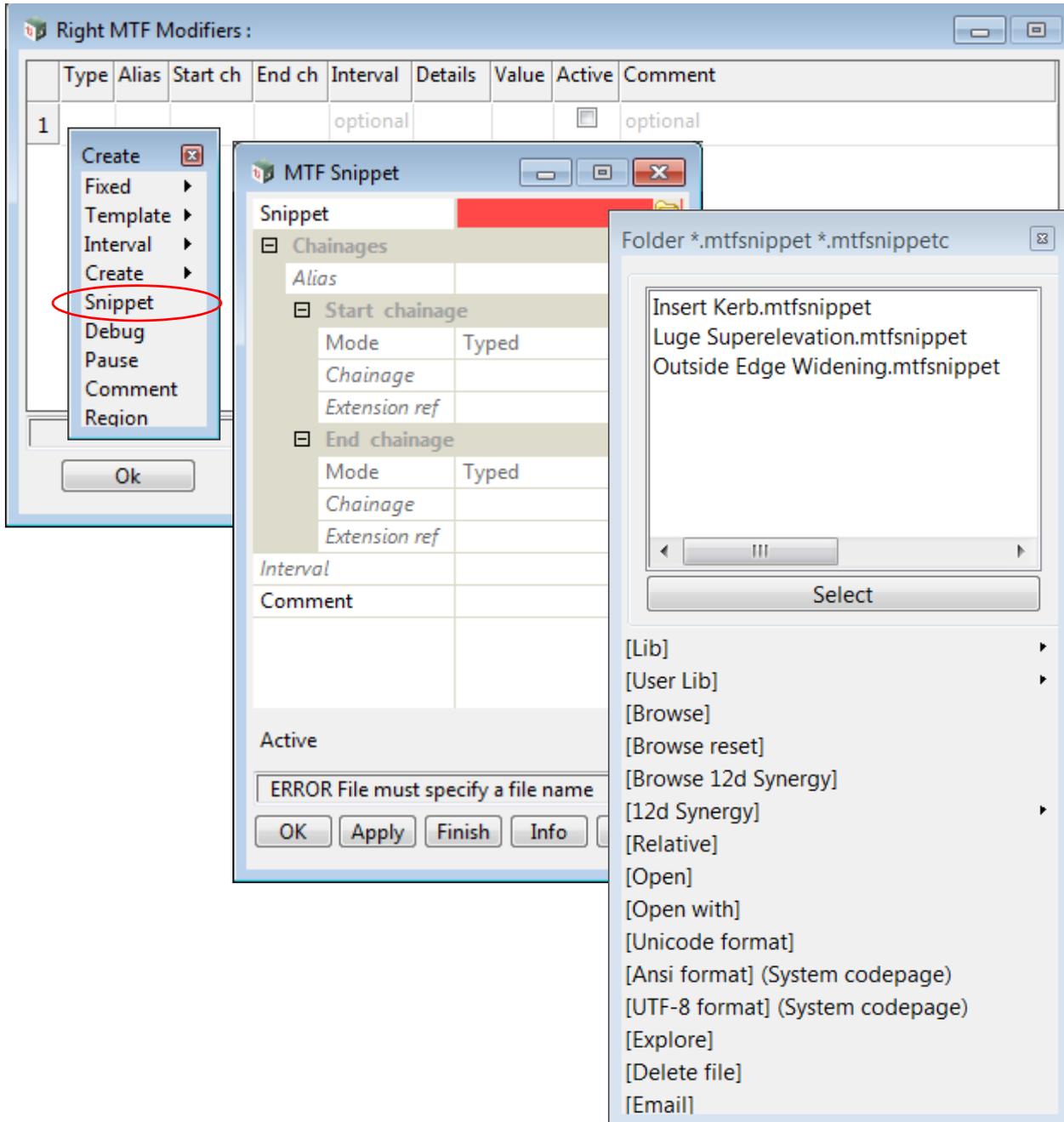
```
// drop into DW
insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0 named_position "MODIFIER_START" 0.5 absolute extra_start extra_end
```

But that would be impossible to read.

Go to the next section [24.13.4 How To Select a Snippet?](#) or back to [24.13 Defining and Using Snippets](#).

## 24.13.4 How To Select a Snippet?

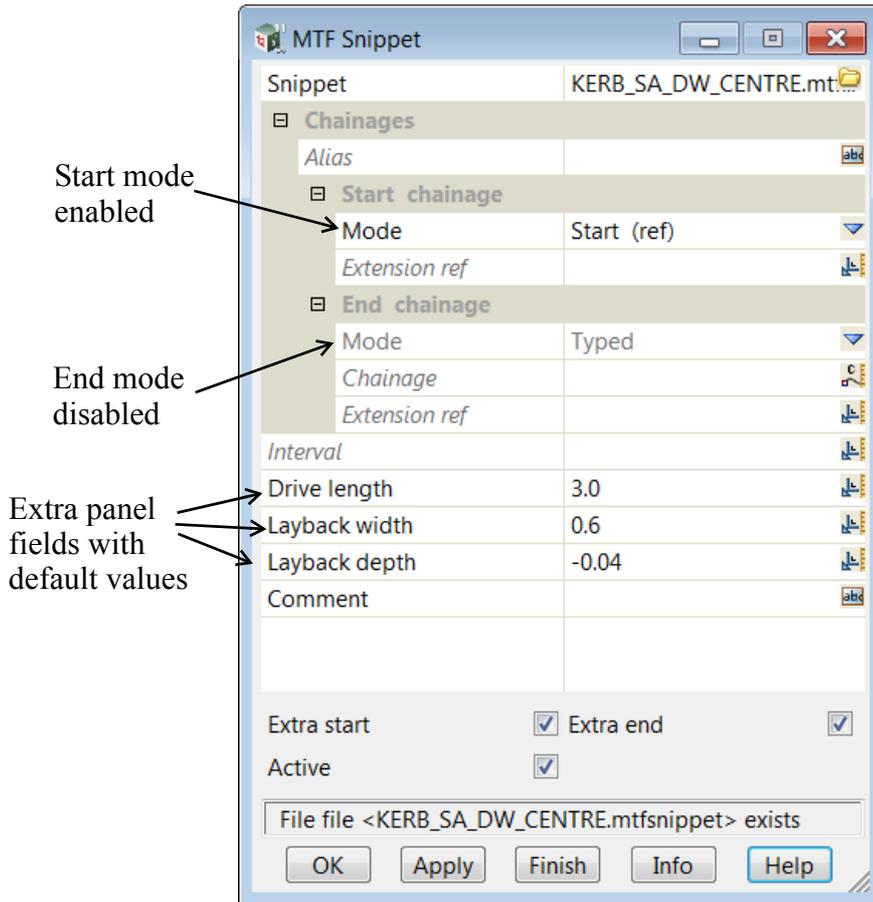
From the grid in the **Right MTF Modifiers** or **Left MTF Modifiers** panel, bring up the **Create** menu and select **Snippet**. This will display the **MTF Snippet** panel and clicking on the folder icon in the **Snippet** field brings up the choice of snippets to use.



A snippet is then selected from the snippet pop-up list.

Selecting a snippet can modify the fields on the **MTF Snippet** panel. The **Start** and **End modes** may or may not be enabled, and there may be extra field on the panel, with or without values in them.

For example, selecting the *KERB\_SA\_DW* snippet modifies the **MTF Snippet** panel and there are two extra panel fields - **Layback width** with the value 0.6 and **Layback height** with the value - 0.4. Selecting the *KERB\_SA\_DW\_CENT* snippet adds the extra panel fields but also disables the **End mode**.

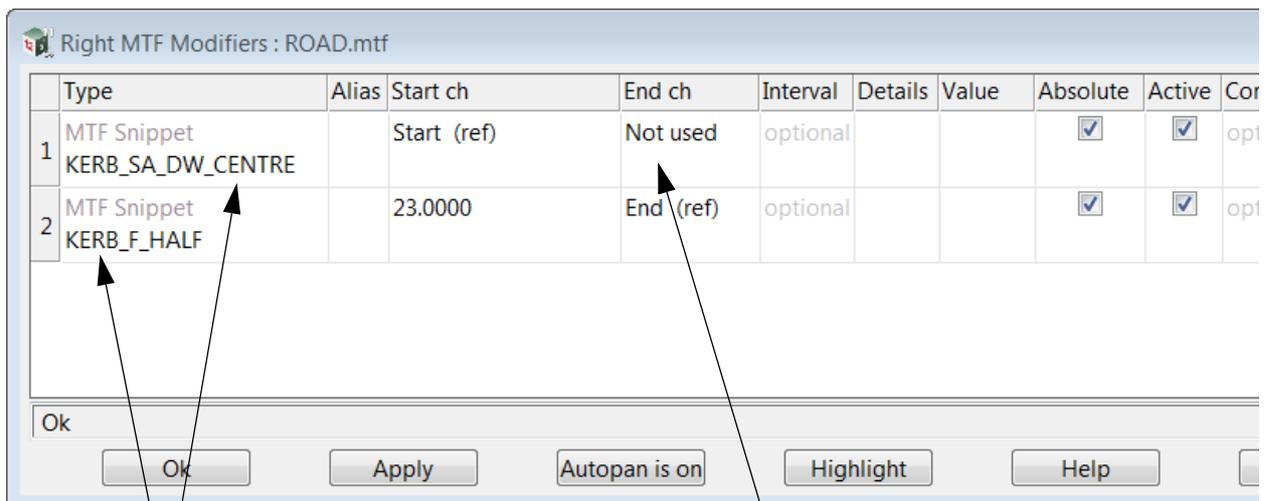


Start mode enabled

End mode disabled

Extra panel fields with default values

If a Start mode or End mode is missing from the snippet, then the corresponding field will be left blank in the **Start chainage/End chainage** columns in the **Left or Right Templates Modifiers** panel. And the snippet name appears in the **Type** column under the text **MTF Snippet**.



Names of the snippets

A snippet with no End mode has **Not used** in the End chainage column.

How snippets control the **Start** and **End** modes is documented in the next section [24.13.6 Start and End Modes for a Snippet](#) and how the extra panel fields are created is discussed in the

sections [24.13.7.1 Snippet Parameter of Type REAL](#), [24.13.7.5 Snippet Parameter of Type CHOICE](#) and [24.13.7.7 Snippet Parameter of Type TICK](#).

Go to the next section [24.13.5 Comments in Snippets](#) or back to [24.13 Defining and Using Snippets](#).

## 24.13.5 Comments in Snippets

Snippets support the same commenting syntax as MTF files, which has two forms:

### 1. A Line Comment

All characters from a double forward-slash `//` until the end of that line are ignored.

However, if the first word after the line comment character is `PARAMETER`, `INFO` or `DISPLAY` then the line will be interpreted as a special command by 12d.

```
// PARAMETER this will not be interpreted as a normal line comment
// This is a normal line comment
@ def_tok WIDTH 3.5 // This is also a line comment
```

### 2. A Block Comment

All characters between a starting `/*` and a terminating `*/` are ignored.

```
/* This comment can
   cover several lines
   until we get to the terminating */
```

Continue to the next section [24.13.6 Start and End Modes for a Snippet](#) or return to [24.13 Defining and Using Snippets](#).

## 24.13.6 Start and End Modes for a Snippet

The **Start** and **End** modes on the **MTF Snippet** panel calculate a position on the Alignment just like any other *MTF modifier* but how the values are specified inside the snippet file is slightly different to a standard MTF modifier.

Also unlike other MTF modifiers, some snippets only require *one* of the **Start mode** and **End mode** and if that is the case, the mode not required will be greyed out so that it can't be filled in.

The lines in the snippet file are similar to the normal MTF modifiers when "named position" has been selected for the **Start** and **End** modes. However the value of the "named position" in the text command needs to be replaced by either MODIFIER\_START or MODIFIER\_END followed by a real number.

So for example, the "named\_position" "name" "extension" in a MTF command is replaced by

```
"named_position" MODIFIER_START snippet_extension
```

If a MODIFIER\_START exists somewhere in the snippet then **Start mode** is enabled on the **MTF Snipped** panel.

If a MODIFIER\_END exists somewhere in the snippet then **End mode** is enabled on the **MTF Snipped** panel.

Finally when a snippet runs, the chainage is calculated for the **Start mode/End mode** fields in the **MTF Snippet** panel and passed down to the snippet and where ever MODIFIER\_START/MODIFIER\_END appears, the passed chainage value PLUS the additional snippet\_extension for each occurrence of the MODIFIER\_START/MODIFIER\_END is used as the chainage for that part of the snippet.

The following example for the KERB\_SA\_DW snippet shows how the snippet determines when a **Start mode** and/or **End mode** is needed., and how the extra snippet\_extension is used.

```
// drop into DW
insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0 named_position
"MODIFIER_START" 0.5 absolute extra_start extra_end

...

// DW to -0.5
insert "SKL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0.5 named_position
"MODIFIER_END" -0.5 absolute extra_start extra_end

...

// back out to -0.5 to end
insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_END" -0.5 named_position
"MODIFIER_END" 0.0 absolute extra_start extra_end
```

These lines in the KERB\_SA\_DW snippet a lip of kerb through a driveway layback. It starts off as an SA style kerb, the layback is an SK style and then back to the SA style.

### 1. Looking at the first line: - the SA->SK transition:

```
insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0 named_position
"MODIFIER_START" 0.5 absolute extra_start extra_end
```

In this line, the **"named\_position" "MODIFIER\_START" 0** replaces the standard chainage command in that position in the text modifier.

**"named\_position" "MODIFIER\_START"** - this tells the snippet to have a **Start mode** on the **MTF Snippet** panel. The calculated chainage for the **Start mode** field on the **MTF Snippet** panel is then passed to the snippet to be used as the chainage (plus the snippet\_extension) in that **"named\_position" MODIFIER\_START** position in the modifier command.

**0** - the value after MODIFIER\_START is an additional value to add to the chainage calculated in from the **Start mode** in the **MTF Snippet** panel and passed down to the snippet.

So **"named\_position" "MODIFIER\_START" 0** is replaced by the chainage defined by the **Start mode** of the **MTF Snippet** panel.

Similarly the **"named\_position" "MODIFIER\_START" 0.5** replaces the standard chainage command in that position in the modifier and evaluates to being chainage defined by the **Start mode** of the **MTF Snippet** panel plus 0.5.

So this snippet line inserts a link from the chainage value passed in from the **Start mode** on the **MTF Snippet** panel, to 0.5 **after** the chainage value passed in from the **Start mode** on the **MTF Snippet** panel.

## 2. Looking at the second line: - SK Layback:

```
insert "SKL" "grey" 0.001 0 unknown named_position "MODIFIER_START" 0.5
named_position "MODIFIER_END" -0.5 absolute extra_start extra_end
```

The difference to the previous line is that this line has the keywords **MODIFIER\_START** and **MODIFIER\_END** after "named\_position" commands.

**"named\_position" "MODIFIER\_END"** - this tells the snippet to have a **End mode** on the **MTF Snippet** panel. The calculated chainage for the **End mode** field on the **MTF Snippet** panel is then passed to the snippet to be used as the chainage (plus the snippet\_extension) in that **"named\_position" MODIFIER\_END** position in the modifier command.

**-0.5** - the value after MODIFIER\_END is an additional value to add to the chainage calculated in from the **End mode** in the **MTF Snippet** panel and passed down to the snippet.

So this snippet line inserts a link from 0.5 past the chainage value passed from **Start mode** in the **MTF Snippet** panel through to 0.5 **before** the chainage value passed in from **End mode** in the **MTF Snippet** panel.

## 3. Looking at the third line: - SK ->SA transition:

```
insert "SAL" "grey" 0.001 0 unknown named_position "MODIFIER_END" -0.5 named_position
"MODIFIER_END" 0.0 absolute extra_start extra_end
```

So this snippet line inserts a link from 0.5 before the chainage value passed to the snippet from the **Start mode** in the **MTF Snippet** panel, to the chainage value passed to the snippet from the **Start mode** in the **MTF Snippet** panel.

Go to the next section [24.13.7 Snippet Parameters](#) or back to [24.13 Defining and Using Snippets](#).

## 24.13.7 Snippet Parameters

A powerful feature of snippets is that parameters of type real, integer and text values, model names, tin names, choices, strings etc can be defined as parameters in the snippet file and the parameters used throughout the snippet file.

To pass values for the parameters through to the snippet, each parameter is displayed as a the appropriate panel Field in the **MTF Snippet** panel, with the descriptive text for the panel field defined in the snippet. For example, a Real box panel field for a real parameter and a Model box panel field for a model parameter.

The value is entered into the panel field in the **MTF Snippet** panel and is then passed through to the snippet.

**Note** - Even if a parameter is defined in a snippet, it will only appear in the MTF Snippet panel if the parameter is actually used in the snippet.

**WARNING** - once a snippet with parameters has been saved in a MTF file, the parameters for the snippet are saved *in the MTF file*. Then when the Apply MTF is run, the parameters and their values are read in from the MTF file and passed into the current snippet file.

Hence if the snippet file has been modified and parameter definitions added, modified or removed since the snippet was originally place in the MTF, these modifications of parameters will be **ignored** until the snippet panel is reopened in the MTF editor which forces the snippet file to be parsed again and the changes to the parameter definitions are then recognised.

For the syntax of each available parameter type, see

- [24.13.7.1 Snippet Parameter of Type REAL](#)
- [24.13.7.2 Snippet Parameter of Type INTEGER](#)
- [24.13.7.3 Snippet Parameter of Type TEXT](#)
- [24.13.7.4 Snippet Parameter of Type SELECT](#)
- [24.13.7.5 Snippet Parameter of Type CHOICE](#)
- [24.13.7.6 Snippet Parameter of Type CHOICE2](#)
- [24.13.7.7 Snippet Parameter of Type TICK](#)
- [24.13.7.8 Snippet Parameter of Type COLOUR](#)
- [24.13.7.9 Snippet Parameter of Type NAME](#)
- [24.13.7.10 Snippet Parameters for Models](#)
- [24.13.7.11 Snippet Parameters for Tins](#)
- [24.13.7.12 Snippet Parameter of Type LAYER](#)
- [24.13.7.13 Snippet Parameter of Type NAMED\\_GRADE](#)
- [24.13.7.14 Snippet Parameter of Type INFO](#)
- [24.13.7.15 Snippet Parameter of Type DISPLAY](#)
- [24.13.7.16 Snippet Parameter of Type INCREMENT](#)
- [24.13.7.17 Optional Parameters in Snippets](#)

### 24.13.7.1 Snippet Parameter of Type REAL

If a floating point number (a real number) is required then the **REAL** parameter type is used. A Snippet Parameter of type **REAL** inserts a Real box panel field into the **MTF Snippet** panel. The user can type a real number into the panel field which is then passed down to the snippet.

The syntax for type **REAL** is:

```
// PARAMETER param_name REAL param_description param_default_value
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**REAL** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

The following lines from the KERB\_SA\_DW snippet define two REAL parameters WIDTH and DEPTH and the values for the parameters are then used inside the snippet commands by writing \$WIDTH and \$DEPTH.

```
// PARAMETER WIDTH REAL "Layback width" 0.6
// PARAMETER DEPTH REAL "Layback depth" -0.04
// drop into DW
...
// push out links
width "SAT" named_position "MODIFIER_START" 0 named_position "MODIFIER_START" 0.5
0.03 $WIDTH absolute extra_start extra_end
...
```

Real panels field are in MTF Snippet panel because of the parameters WIDTH and DEPTH in the snippet

Description provided by the parameter definition in the snippet

Interval: [ ] [ ] [ ]

Layback width: [ 0.6 ] [ ] [ ]

Layback depth: [ -0.04 ] [ ] [ ]

Extra start:  Extra end:

Default value provided by the parameter definition in the snippet

**Important Note:**

The Real panel field for a REAL parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.2 Snippet Parameter of Type INTEGER](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.2 Snippet Parameter of Type INTEGER

If a whole number (an integer) is required then the **INTEGER** parameter type is used.

A Snippet Parameter of type **INTEGER** inserts an Integer box panel field into the **MTF Snippet** panel.

The user can type an integer into the panel field and it is then passed down to the snippet.

The syntax for type **INTEGER** is:

```
// PARAMETER param_name INTEGER param_description param_default_value
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**INTEGER** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclose in double quotes (").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes (").

**Important Note:**

The Integer panel field for a **INTEGER** parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.3 Snippet Parameter of Type TEXT](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.3 Snippet Parameter of Type TEXT

If text is required then the **TEXT** parameter type is used.

A Snippet Parameter of type **TEXT** inserts an Text box panel field into the **MTF Snippet** panel.

The user can type text into the panel field and it is then passed down to the snippet.

The syntax for type **TEXT** is:

```
// PARAMETER param_name TEXT param_description param_default_value
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**TEXT** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quote characters (e.g. "light blue").

*Note for the TEXT type, the \$parameter should be enclosed in the double quotes character in the modifier line.*

**Important Note:**

The Text panel field for a TEXT parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.4 Snippet Parameter of Type SELECT](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.7.4 Snippet Parameter of Type SELECT

If a string is to be selected then the **SELECT** parameter type is used.

A Snippet Parameter of type **SELECT** inserts a String Select box panel field into the **MTF Snippet** panel.

Using the Select box, a user can pick a string and it is then passed down to the snippet.

The syntax for type **SELECT** is:

```
// PARAMETER param_name SELECT param_description param_default_value
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**SELECT** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER BB_STRING SELECT "Base string"
```

...

```
// parallel the selected string
```

```
fixed_parallel "SFML" "SFMB" named_position "MODIFIER_START" 0 named_position  
"MODIFIER_FINAL" 0 "$BB_STRING" interval 0.5 absolute extra_start extra_end
```

...

String select panel field is in MTF Snippet panel because of the SELECT parameter BB\_STRING in the snippet



Description provided by the parameter definition in the snippet

Here the parameter name is **BB\_STRING**, its type is **SELECT** and so a **Select** box appears in the **MTF Snippet** panel with the panel field text **Base string**. The substitution point in the modifier is "**\$BB\_STRING**".

*Note for the SELECT type, the \$parameter should be quoted in the modifier line.*

**Important Note:**

The String Select panel field for a SELECT parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.5 Snippet Parameter of Type CHOICE](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.5 Snippet Parameter of Type CHOICE

If the user is to select from a list of choices and the text of the choice is to be passed through to the snippet, then the **CHOICE** parameter type is used.

The Snippet Parameter of type **CHOICE** inserts a Choice box panel field into the **MTF Snippet** panel.

The format also defines the list of text to be displayed in the pop-up of choices and when a choice is selected, the text that is displayed is passed back to the snippet.

The syntax for type **CHOICE** is:

```
// PARAMETER param_name CHOICE param_description choice_1 choice_2 ... choice_n
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**CHOICE** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes (").

**choice\_1 choice\_2 ... choice\_n** - the choices that will be displayed in the parameter pop-up in the **MTF Snippet** panel.

```
// PARAMETER KT CHOICE "Kerb type" "SA" "SK" "SM" "SC"

insert "$KT" "grey" 10 0 unknown named_position "MODIFIER_START" (0.0 - _E)
named_position "MODIFIER_END" _E absolute extra_start extra_end

...
```

Choice box panel field is in the MTF Snippet panel because of the parameter KT of type TICK

Description provided by the parameter definition in the snippet

Pop-up of choices from the parameter definition in the snippet

**Important Note:**

The Choice panel field for a CHOICE parameter will only appear in the **MTF Snippet** panel if the

parameter is used in the snippet.

Continue to the next section [24.13.7.6 Snippet Parameter of Type CHOICE2](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.6 Snippet Parameter of Type CHOICE2

If the user is to select from a list of choices, but different text than what appears on the choice list is to be passed through to the snippet, then the **CHOICE2** parameter type is used.

The Snippet Parameter of type **CHOICE2** inserts a Choice box panel field into the **MTF Snippet** panel.

The format also defines the list of text to be displayed in the pop-up of choices for each text that is displayed, a parameter value that is passed back to the snippet if that choice is selected.

The syntax for type **CHOICE2** is:

```
// PARAMETER par_name CHOICE2 par_desc par_def par_1 desc_1 par_2 desc_2 ... par_n desc_n
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**par\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$par\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**CHOICE2** - the parameter type.

**par\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclose in double quotes (").

**par\_def** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes (").

**par\_i desc\_i** are pairs of parameters and description to go in the Choice box pop up. The description **desc\_i** is displayed in the Choice pop up and if that description is selected, the parameter **par\_i** is returned to the Snippet instead of the description.

For example

```
// PARAMETER CH CHOICE2 "Speed" S100 S80 "80 km/h" S100 "100 km/h"
```

This means that the text you have the default parameter, in this case **S100**, and then pairs of parameters and descriptions, **S80 "80 km/h"** etc.

```
// PARAMETER CH CHOICE2 "Layer" "Design" "Design" "Surface of the road" "Subbase" "Subbase layer"  
insert $CH=>$LINK" "$CLR" 0.001 0.20 unknown SCH 0.0 ECH 0.0 absolute extra_start extra_end
```

...

Choice box panel field is in the MTF Snippet panel because of the parameter CH of type CHOICE2

New link		N
Colour	red	
Layer	Surface of the road	
Comment		abr

Pop-up of choices from the parameter definition in the snippet

Select Choice 

- Surface of the road
- Subbase layer

Description provided by the parameter definition in the snippet

**Important Note:**

The Choice panel field for a CHOICE2 parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.7 Snippet Parameter of Type TICK](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.7 Snippet Parameter of Type TICK

If a true or false choice is required then the **TICK** parameter type is used.

The Snippet Parameter of type **TICK** inserts a Tick box panel field into the **MTF Snippet** panel.

The user can set the tick to off or on.

The syntax for type TICK is:

```
// PARAMETER param_name TICK param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by case are considered to be identical. The parameters names must be unique in a snippet.

**TICK** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then it must be 1 or a 0.

If 1, the tick box comes up set to tick/on.

If 0, the tick box comes up set to not ticked/off.

```
// PARAMETER KT CHOICE "Kerb type" "SA" "SK" "SM" "SC"
// PARAMETER TI TICK "Use extensions" 1
#if $TI
#define _E 3
#else
#define _E 0
#endif

insert "$KT" "grey" 10 0 unknown named_position "MODIFIER_START" (0.0 - _E)
named_position "MODIFIER_END" _E absolute extra_start extra_end
```

...

Tick box panel field is in the MTF Snippet panel because of the parameter TI of type TICK

Description provided by the parameter definition in the snippet



Choice of tick and not tick because the parameter type is TICK

Note that in the above example, the value of the Tick box parameter TI (\$TI), is used in a #define to set the value for **\_E** (see [24.13.17 Major Warning - You Will be Caught by This](#)).

**Important Note:**

The Tick panel field for a TICK parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.8 Snippet Parameter of Type COLOUR](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.8 Snippet Parameter of Type COLOUR

If a colour is required then the **COLOUR** parameter type is used.

The Snippet Parameter of type **COLOUR** inserts a Colour box panel field into the **MTF Snippet** panel.

The user can type a colour name or colour number into the panel field, or select a colour from the pop-up of available colours in the project.

The syntax for type **COLOUR** is:

```
// PARAMETER param_name COLOUR param_description param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**COLOUR** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

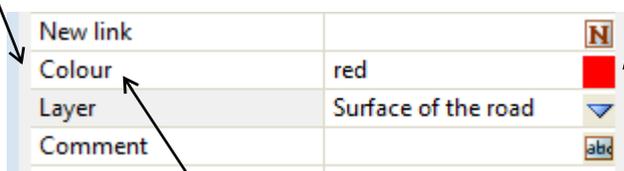
**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER CLR COLOUR "Colour" "red"
insert "$INS_LYR=>$LINK" "$CLR" 0.001 0.20 unknown SCH 0.0 ECH 0.0 absolute extra_start extra_end
```

...

Colour box panel field is in the MTF Snippet panel because of the parameter CLR of type COLOUR

Pop-up of colours because the parameter type is COLOUR



Description provided by the parameter definition in the snippet

**Important Note:**

The Colour panel field for a COLOUR parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.9 Snippet Parameter of Type NAME](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.7.9 Snippet Parameter of Type NAME

If a name of a string is required then the **NAME** parameter type is used.

The Snippet Parameter of type **NAME** inserts a Name box panel field into the **MTF Snippet** panel.

The user can type a string name into the panel field, or select a name from the pop-up of names that are taken from the names.4d file.

The syntax for type **NAME** is:

```
// PARAMETER param_name NAME param_description param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**NAME** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

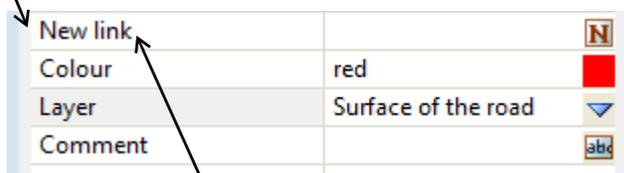
**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER LINK NAME "New link"
// PARAMETER CLR COLOUR "Colour" "red"
insert "$INS_LYR=>$LINK" "$CLR" 0.001 0.20 unknown SCH 0.0 ECH 0.0 absolute extra_start extra_end
```

...

Name box panel field is in the MTF Snippet panel because of the parameter LINK of type NAME

Pop-up of names because the parameter type is NAME



Description provided by the parameter definition in the snippet

**Important Note:**

The String Name panel field for a NAME parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.10 Snippet Parameters for Models](#) or return to [24.13.7](#)

[Snippet Parameters](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.7.10 Snippet Parameters for Models

There are three different Snippet parameter types for creating a Model box in the **MTF Snippet** panel. Which parameter is used depends on what is required to happen if the model exists or does not exist.

- (a) use MODEL\_CREATE when it doesn't matter if the selected model exists or doesn't exist.

See [24.13.7.10.1 Snippet Parameters of Type MODEL\\_CREATE](#)

- (b) use MODEL\_MUST\_EXIST when the selected model must already exist.

See [24.13.7.10.2 Snippet Parameter of Type MODEL\\_MUST\\_EXIST](#)

- (c) use MODEL\_MUST\_NOT\_EXIST when the model must **not** already exist.

[24.13.7.10.3 Snippet Parameter Type MODEL\\_MUST\\_NOT\\_EXIST](#)

Or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.10.1 Snippet Parameters of Type MODEL\_CREATE

The Snippet Parameter of type **MODEL\_CREATE** inserts a Model box panel field into the **MTF Snippet** panel.

The user can type a model name into the panel field, or select a model from the pop-up of models for the Model box.

If the model does not exist when the snippet is run, then a new model is created.

The syntax for type **MODEL\_CREATE** is:

```
// PARAMETER param_name MODEL_CREATE param_desc param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**MODEL\_CREATE** - the parameter type.

**param\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER TM MODEL_CREATE "Trimesh model" "Tri RSK 0201"
create_strings "selection" "KSA<<LK" "KSA<<IK" "KSA<<TK" "KSA<<BK" start_ref 0 final_ref 0
"no_colour" "" "$TM" absolute extra_start extra_end

...

```

Model box panel field is in the MTF Snippet panel because of the parameter TM of type MODEL\_CREATE

Pop-up of models because the parameter type is MODEL\_CREATE

Colour	red
Layer	Surface of the road
Trimesh model	Tri RSK 0201

Description provided by the parameter definition in the snippet

**Important Note:**

The Model panel field for a MODEL\_CREATE parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.10.2 Snippet Parameter of Type MODEL\\_MUST\\_EXIST](#) or

return to [24.13.7.10 Snippet Parameters for Models](#) or [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.7.10.2 Snippet Parameter of Type MODEL\_MUST\_EXIST

The Snippet Parameter of type **MODEL\_MUST\_EXIST** inserts a Model box panel field into the **MTF Snippet** panel.

The user can type a model name into the panel field, or select a model from the pop-up of models for the Model box.

If the model does **not** exist when the Snippet is run, an error will occur and a message written to the panel message area.

The syntax for type **MODEL\_MUST\_EXIST** is:

```
// PARAMETER param_name MODEL_MUST_EXIST param_desc param_default
where
```

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**MODEL\_MUST\_EXIST** - the parameter type.

**param\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclose in double quotes ("").

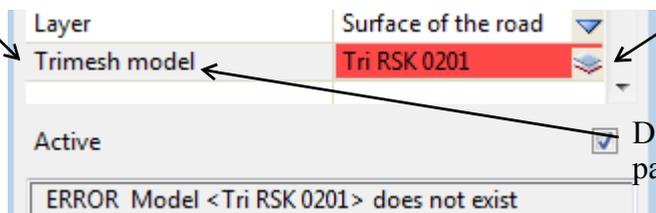
**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER TM MODEL_MUST_EXIST "Trimesh model" "Tri RSK 0201"
create_strings "selection" "KSA<<LK" "KSA<<IK" "KSA<<TK" "KSA<<BK" start_ref 0 final_ref 0
    "no_colour" "" "$TM" absolute extra_start extra_end
...

```

Model box panel field is in the MTF Snippet panel because of the parameter TM of type MODEL\_MUST\_EXIST

Pop-up of models because the parameter type is MODEL\_MUST\_EXIST



Description provided by the parameter definition in the snippet

Error message if the model does not exist

**Important Note:**

The Model panel field for a MODEL\_MUST\_EXIST parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.10.3 Snippet Parameter Type MODEL\\_MUST\\_NOT\\_EXIST](#) or return to [24.13.7.10 Snippet Parameters for Models](#) or [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.7.10.3 Snippet Parameter Type MODEL\_MUST\_NOT\_EXIST

The Snippet Parameter of type **MODEL\_MUST\_NOT\_EXIST** inserts a Model box panel field into the **MTF Snippet** panel.

The user can type a model name into the panel field, or select a model from the pop-up of models for the Model box.

If the model **does** exist when the Snippet is run, an error will occur and a message written to the panel message area.

The syntax for type **MODEL\_MUST\_NOT\_EXIST** is:

```
// PARAMETER param_name MODEL_MUST_NOT_EXIST param_desc param_default
where
```

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**MODEL\_MUST\_NOT\_EXIST** - the parameter type.

**param\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclose in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER TM MODEL_MUST_NOT_EXIST "Trimesh model" "Tri RSK 0201"
create_strings "selection" "KSA<<LK" "KSA<<IK" "KSA<<TK" "KSA<<BK" start_ref 0 final_ref 0
    "no_colour" "" "$TM" absolute extra_start extra_end
...
Model box panel field is in the
MTF Snippet panel because of
the parameter TM of type
MODEL_MUST_NOT_EXIST
Pop-up of models because
the parameter type is
MODEL_MUST_NOT_EXIST
Description provided by the
parameter definition in the snippet
ERROR Model <RS K> exists
Error message if the model already exists
```

**Important Note:**

The Model panel field for a MODEL\_MUST\_NOT\_EXIST parameter will only appear in the **MTF**

**Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.11 Snippet Parameters for Tins](#) or return to [24.13.7.10 Snippet Parameters for Models](#) or [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.11 Snippet Parameters for Tins

There are three different Snippet parameter types for creating a Tin box in the **MTF Snippet** panel. Which parameter is used depends on what is required to be done if the tin exists or does not exist.

- (a) use TIN\_CREATE when it doesn't matter if the selected tin exists or doesn't exist.

See [24.13.7.11.1 Snippet Parameter Type TIN\\_CREATE](#)

- (b) use TIN\_MUST\_EXIST when the selected model must already exist.

See [24.13.7.11.2 Snippet Parameter Type TIN\\_MUST\\_EXIST](#)

- (c) use TIN\_MUST\_NOT\_EXIST when the model must **not** already exist.

[24.13.7.11.3 Snippet Parameter Type TIN\\_MUST\\_NOT\\_EXIST](#)

Or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.7.11.1 Snippet Parameter Type TIN\_CREATE

The Snippet Parameter of type **TIN\_CREATE** inserts a Tin box panel field into the **MTF Snippet** panel.

The user can type a tin name into the panel field, or select a tin from the pop-up of tins for the Tin box.

If the tin does not exist when the snippet is run, then a new tin is created.

The syntax for type **TIN\_CREATE** is:

```
// PARAMETER param_name TIN_CREATE param_desc param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**TIN\_CREATE** - the parameter type.

**param\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER TIN TIN_CREATE "Tin to intersect" "Ground"
tin_all_fixed "Design<<BA" start_ref 0 final_ref 0 "$TIN" "mod_wdt_hgt_fix" 1 absolute extra_start extra_end

...
```

Tin box panel field is in the MTF Snippet panel because of the parameter TIN of type TIN\_CREATE

Pop-up of tins because the parameter type is TIN\_CREATE



Description provided by the parameter definition in the snippet

#### Important Note:

The Tin panel field for a TIN\_CREATE parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.11.2 Snippet Parameter Type TIN\\_MUST\\_EXIST](#) or return

to [24.13.7.11 Snippet Parameters for Tins](#) or [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.11.2 Snippet Parameter Type TIN\_MUST\_EXIST

The Snippet Parameter of type **TIN\_MUST\_EXIST** inserts a Tin box panel field into the **MTF Snippet** panel.

The user can type a tin name into the panel field, or select a tin from the pop-up of tins for the Tin box.

If the tin does **not** exist when the Snippet is run, an error will occur and a message written to the panel message area.

The syntax for type **TIN\_MUST\_EXIST** is:

```
// PARAMETER param_name TIN_MUST_EXIST param_desc param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**TIN\_MUST\_EXIST** - the parameter type.

**param\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER TIN TIN_MUST_EXIST "Tin to intersect" "Ground"
tin_all_fixed "Design<<BA" start_ref 0 final_ref 0 "$TIN" "mod_wdt_hgt_fix" 1 absolute extra_start extra_end
...
Tin box panel field is in the
MTF Snippet panel because of
the parameter TIN of type TIN_MUST_EXIST
```

Pop-up of tins because the parameter type is TIN\_MUST\_EXIST

Description provided by the parameter definition in the snippet

Error message if the tin does not exist

**Important Note:**

The Tin panel field for a **TIN\_MUST\_EXIST** parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.11.3 Snippet Parameter Type TIN\\_MUST\\_NOT\\_EXIST](#) or return to [24.13.7.11 Snippet Parameters for Tins](#) or [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.11.3 Snippet Parameter Type TIN\_MUST\_NOT\_EXIST

The Snippet Parameter of type **TIN\_MUST\_NOT\_EXIST** inserts a Tin box panel field into the **MTF Snippet** panel.

The user can type a tin name into the panel field, or select a tin from the pop-up of tins for the Model box.

If the tin **does** exist when the Snippet is run, an error will occur and a message written to the panel message area.

The syntax for type **TIN\_MUST\_NOT\_EXIST** is:

```
// PARAMETER param_name TIN_MUST_NOT_EXIST param_desc param_default
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$param\_name**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**TIN\_MUST\_NOT\_EXIST** - the parameter type.

**param\_desc** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclose in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

```
// PARAMETER TIN TIN_MUST_NOT_EXIST "Tin to intersect" "Ground"
tin_all_fixed "Design<<BA" start_ref 0 final_ref 0 "$TIN" "mod_wdt_hgt_fix" 1 absolute extra_start extra_end

...
Tin box panel field is in the
MTF Snippet panel because of
the parameter TIN of type
TIN_MUST_NOT_EXIST
```

Pop-up of tins because the parameter type is TIN\_MUST\_NOT\_EXIST

Description provided by the parameter definition in the snippet

Error message if the tin already exists

**Important Note:**

The Tin panel field for a **TIN\_MUST\_NOT\_EXIST** parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.12 Snippet Parameter of Type LAYER](#) or return to [24.13.7.11 Snippet Parameters for Tins](#) or [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.12 Snippet Parameter of Type LAYER

If a layer for links is required then the **LAYER** parameter type is used.

A Snippet Parameter of type **LAYER** inserts an Layer box panel field into the **MTF Snippet** panel.

The user can type a layer name into the panel field or select a layer name from the pop-up of layers previously defined. The Layer name is then passed down to the snippet.

The syntax for type **LAYER** is:

```
// PARAMETER param_name LAYER param_description param_default_value
```

where

// - this is needed at the beginning of the line

**PARAMETER** - this is needed after the "/"

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$(param\_name)**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**LAYER** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes ("").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default layer value does not already exist, it will be added to the choice list of Layer names. If the default value is text and it contains any spaces then the text must be enclosed in double quotes ("").

**Important Note:**

The Layer panel field for a LAYER parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.13 Snippet Parameter of Type NAMED\\_GRADE](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.13 Snippet Parameter of Type NAMED\_GRADE

If a **named grade definition** is required then the **NAMED\_GRADE** parameter type is used.

A Snippet Parameter of type **NAMED\_GRADE** inserts an Named Grade box panel field into the **MTF Snippet** panel.

The user can type a named grade into the panel field or select a named grade from the pop-up of named grades previously defined. The Named Grade is then passed down to the snippet.

The syntax for type **NAMED\_GRADE** is:

```
// PARAMETER param_name NAMED_GRADE param_description param_default_value
```

where

```
// - this is needed at the beginning of the line
```

```
PARAMETER - this is needed after the "/"
```

**param\_name** - this is the name to use for the parameter. Where ever the parameter is to be used in the snippet, put **\$(param\_name)**. The characters of the parameter name can only be alphanumeric (upper and lower case) and underscores, and the name cannot contain spaces. The name doesn't have quotes around it. The parameter names are not case sensitive so two names only differing by chase are considered to be identical. The parameters names must be unique in a snippet.

**NAMED\_GRADE** - the parameter type.

**param\_description** - this description is written in the **MTF Snippet** panel for the field to enter values for this parameter. The description can include spaces and if it does then the description must be enclosed in double quotes (").

**param\_default** - this value is **optional** but if it exists then this value is displayed as the value in the field for the parameter on the **MTF Snippet** panel. If the default layer value does not already exist, it will be added to the choice list of Layer names. If the default value is text and it contains any spaces then the text must be enclosed in double quotes (").

**Important Note:**

The Named Grade panel field for a **NAMED\_GRADE** parameter will only appear in the **MTF Snippet** panel if the parameter is used in the snippet.

Continue to the next section [24.13.7.14 Snippet Parameter of Type INFO](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

### 24.13.7.14 Snippet Parameter of Type INFO

There is an **Info** button on the **MTF Snippet** panel and when clicked, all the lines of the snippet file that start with `// INFO` are displayed in **Snippet Description** pop up.

The syntax for **INFO** is:

```
// INFO text_to_display
```

where

`//` - this is needed at the beginning of the line

**INFO** - this is needed after the `//`

**text\_to\_display** - this is a line of text that is to be displayed in the **Snippet Description** pop up when the **Info** button is clicked on the **MTF Snippet** panel. The text is taken from one space after the word **INFO** and to the end of the line.

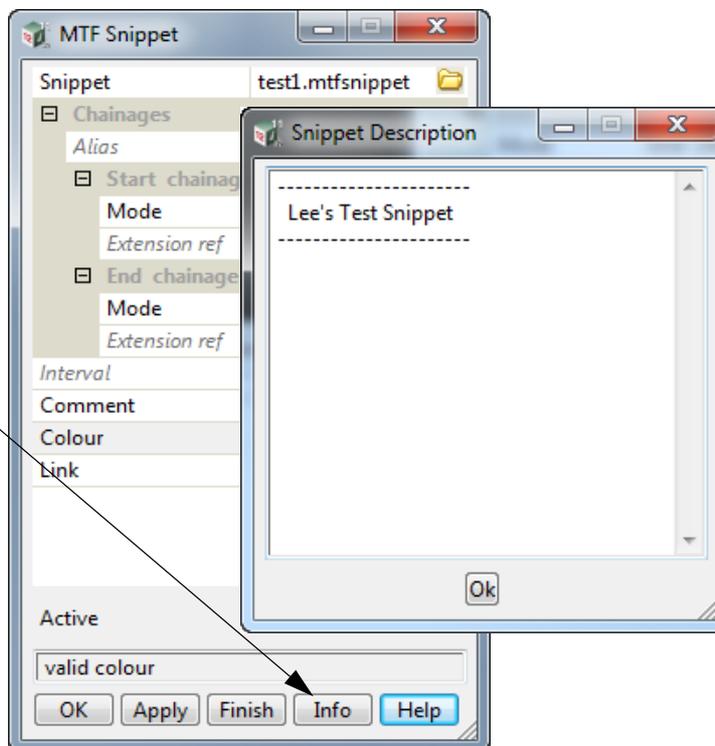
For example, in the Snippet code:

```
// PARAMETER CLR COLOUR "Colour" "red"
// PARAMETER LN TEXT "Link" ""
// INFO -----
// INFO Lee's Test Snippet
// INFO -----
```

← INFO commands

```
insert "Design=>$(LN)" $CLR 0.0 0.0 unknown $_SCH) 0 $_ECH) 0 absolute extra_start extra_end
```

Click on **Info** and the **Snippet Description** pop-up displays all the **INFO** lines in the Snippet.



Continue to the next section [24.13.7.15 Snippet Parameter of Type DISPLAY](#) or return to [24.13 Defining and Using Snippets](#).

### 24.13.7.15 Snippet Parameter of Type DISPLAY

The **parameter description** and the **current value** that the parameter has can be displayed in the **Detail** and **Value** columns of the **MTF Modifiers** panel by using the **DISPLAY** command in the snippet.

The syntax for **DISPLAY** is:

```
// DISPLAY param_name
```

where

// - this is needed at the beginning of the line

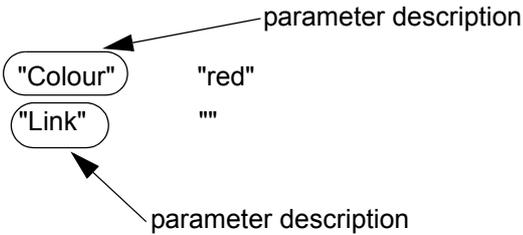
**DISPLAY** - this is needed after the "//"

**param\_name** - this is the name of the parameters to have its description and current value displayed in the **MTF Modifiers** panel.

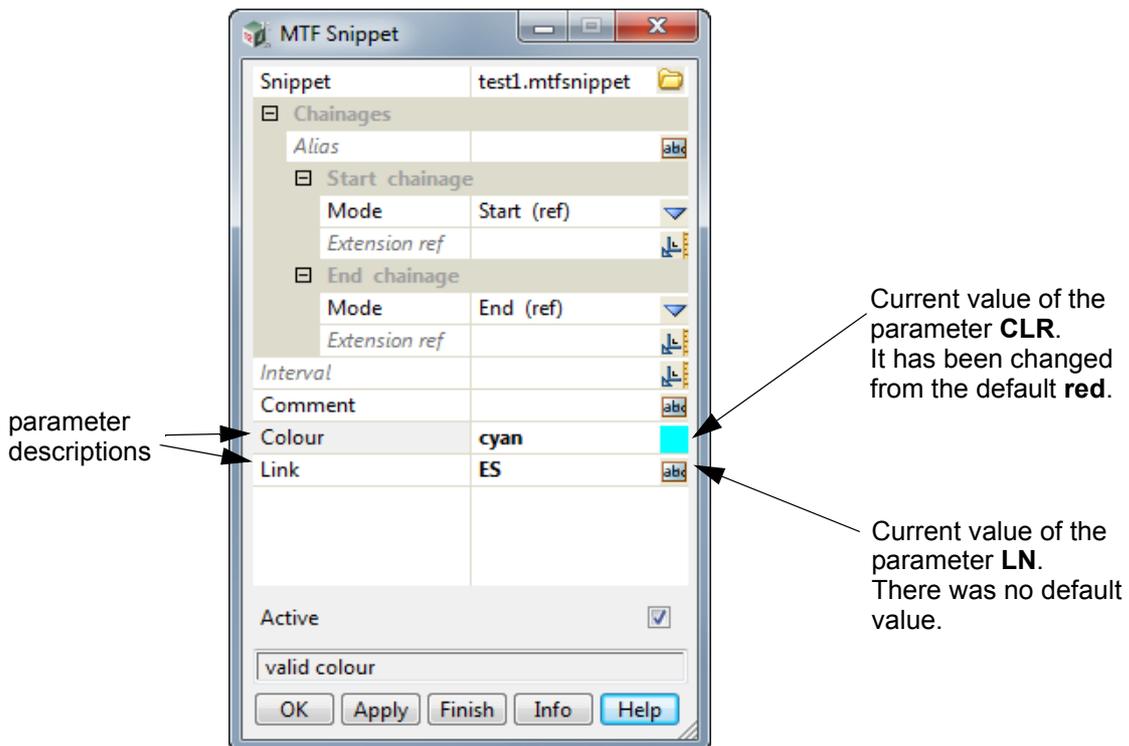
When the **OK** or **Apply** button is clicked on the **MTF Snippets** panel, the **description** of the parameter **param\_name** is displayed in the **Details** column of the **MTF Modifiers** panel and the **current value** of the parameter **param\_name** is displayed in the **Value** column of the **MTF Modifiers** panel.

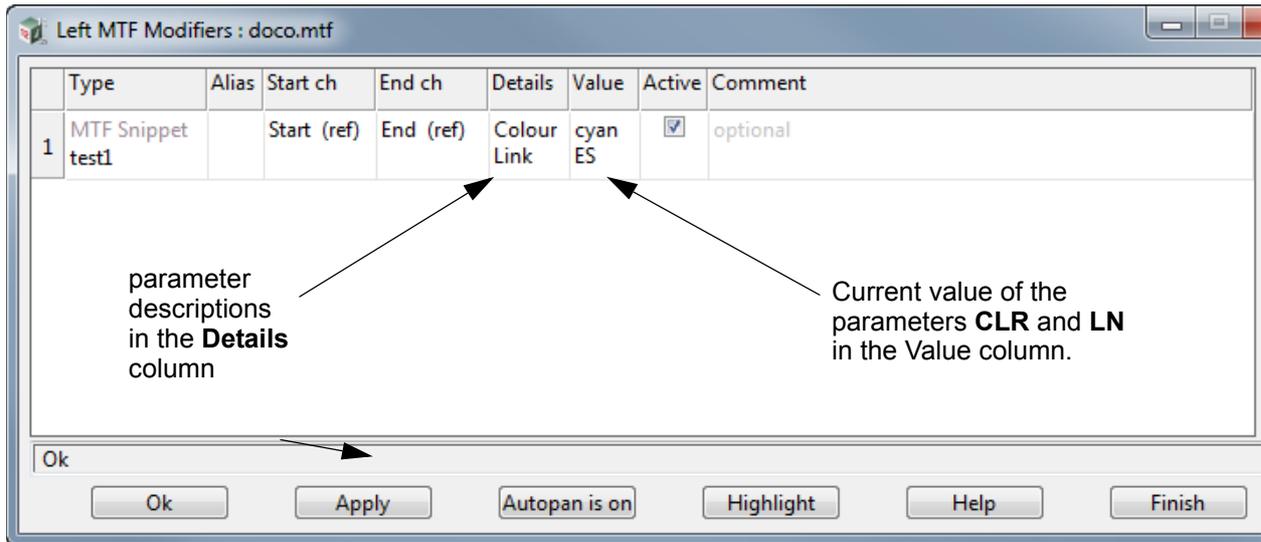
For example, in the Snippet code:

```
// PARAMETER CLR COLOUR "Colour" "red"
// PARAMETER LN TEXT "Link" ""
// DISPLAY CLR
// DISPLAY LN
```



```
insert "Design=>$(LN)" $CLR 0.0 0.0 unknown $(_SCH) 0 $(_ECH) 0 absolute extra_start extra_end
```





Continue to the next section [24.13.14.3 Temporary MTF Snippet File](#) or return to [24.13 Defining and Using Snippets](#).

### 24.13.7.16 Snippet Parameter of Type INCREMENT

There is an **INCREMENT** command to increment a snippet parameter by a given increment step. The increment step can be negative for a decrement.

The intended use is for TEXT parameters but it will actually work for any type of parameter but probably in an unexpected way.

How a parameter is incremented depends on what the value of the parameter is when it is considered as text. We will refer to the value of the parameter as text, as **value\_text**.

For example, for an INTEGER, the **value\_text** could be "13", for a TEXT it could be "STN003", for a REAL, it could be "23.45" and for a COLOUR, it could be "red". Obviously it usually doesn't make sense to increment some parameters like REAL or COLOUR but it can be done.

The INCREMENT command only increments either a

(a) a group of digits at the end of the text. That is each character is one of "0" to "9". The number of digits to increment is fixed (the length of the group).

or

(b) a group of letters at the end of the text. That is, each characters is one of "a" to "z". The number of letters to increment is fixed (the length of the group). Upper and lower case letters are considered to be the same thing.

Note that INCREMENT does not increment a mixture of digits and letters.

For the **value\_text**, the group of characters at the end of the **value\_text** that will be incremented is determined by looking at the last character of the **value\_text** and

(a) if the last character is a **digit** then it will be a **digit group**.

The rest of **value\_text** is then searched from right to left until a letter is found. So all the digits at the end of the **value\_text** make up a **digit group** of a fixed length.

For example, for "STN001" the group is digit group of length 3 ("001").

For "ST2N02", the group is a digit group of length 2 ("02").

For "002", the group is a digit group of length 3 ("002").

The digit group increments by the increment step but it will always keep the same number of digits. So if the incremented digit group would need an extra digit then that extra leading digit is dropped. So the number wraps around.

For example, incrementing "568" by 1 will eventually reach "999" and the next increment will give you "000".

(b) if the last character is an letter (i.e. one of a to z) then it will be a **letter group**.

The rest of **value\_text** is then searched from right to left until a digit is found. So all the letters at the end of **value\_text** make up a **letter group** of a fixed length.

For example, for "123ABC" the group is an letter group of length 3 ("ABC").

"incrementing a letter" means that you take the letters position in the alphabet and add the increment step to it to give you the position of the new incremented letter. For example, **c** is the third letter and increment it by two give the fifth letter which is **e**. When you have a letter group then you work in base 26. For example, incrementing AZ by 1 gives you BA.

As in the case for a digit group, the letter group increments by the increment step but it will always keep the same number of letters. So if the incremented digit group would need an extra letter then that extra leading letter is dropped.

For example, incrementing "TUV" by 1 will eventually reach "ZZZ" and the next increment will give you "AAA".

The syntax for **INCREMENT** is:

```
// INCREMENT param_name increment_step
```

where

// - this is needed at the beginning of the line

**INCREMENT** - this is needed after the "//"

**param\_name** - the name of the parameter to increment.

**increment\_step** - this is the amount the increment the group of digits or letters. The increment step can be negative for a decrement.

As an example,

```
// PARAMETER STRING_ID TEXT "String number" 1
```

```
// INCREMENT STRING_ID 1
```

Continue to the next section [24.13.7.17 Optional Parameters in Snippets](#) or return to [24.13 Defining and Using Snippets](#).

### 24.13.7.17 Optional Parameters in Snippets

Parameters can be set to be optional so that when they are displayed in the **MTF Snippet** panel and there is no value in the panel field, the Description is greyed out.

A parameters is made optional by adding **OPTIONAL** as the last word on the parameter line.

A default value for the parameter can still be passed in but in that case the Description in the **MTF Snippet** panel field will not be greyed out when it is first displayed because there is a value in it.

For example, in the Snippet code:

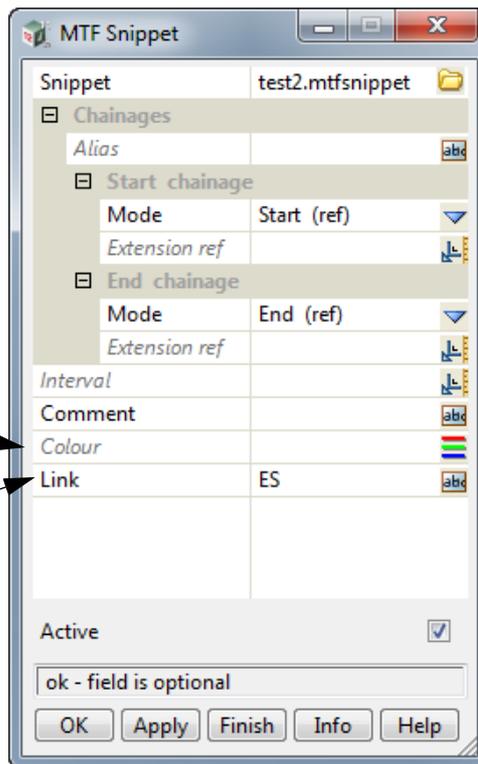
```
// PARAMETER CLR COLOUR "Colour" OPTIONAL
// PARAMETER LN TEXT "Link" "ES" OPTIONAL
// DISPLAY CLR
// DISPLAY LN
insert "Design=>$(LN)" $CLR 0.0 0.0 unknown $_(SCH) 0 $_(ECH) 0 absolute extra_start extra_end
```

the default value has been omitted

default value

Colour is optional

Link is optional but because there is a value in it, the word Link is not greyed out



**WARNING**

Be careful when making a parameter optional because the MTF may then have errors.

For example in the above example, if the **Colour** is not filled in then there will be an error in the MTF because the command

insert "Design=>\$(LN)" \$CLR 0.0 0.0 unknown \$\_(SCH) 0 \$\_(ECH) 0 absolute extra\_start extra\_end will not be valid if CLR is blank and hence not there.

Continue to the next section [24.13.8 Automatic Parameters in Snippets](#) or return to [24.13.7 Snippet Parameters](#) or [24.13 Defining and Using Snippets](#).

## 24.13.8 Automatic Parameters in Snippets

In addition to the ability to create and define your own parameters in snippets, **12d Model** has a range of predefined parameters that are automatically created and calculated.

Automatic parameters allow for the use of values that are important to the snippet, but that you do not necessarily want to prompt the user to complete.

For example, string naming that is dependent on whether it is applied in the left side or right side modifiers. Rather than making the user select the correct left/right side value as a parameter, an automatic parameter can be used instead. In this way, automatic parameters can greatly simplify snippets and make them more portable.

Automatic parameters are fixed by the system and are identified by **starting with an underscore**, e.g. `$_AUTO_LR` or `$_AUTO_LR`.

All automatic parameter names are reserved in 12d. To ensure future compatibility, user-defined parameters should not start with an underscore.

Currently, the following automatic parameters are defined:

[24.13.8.1 \\_AUTO\\_LR](#)

[24.13.8.2 \\_SCH](#)      [24.13.8.3 \\_ECH](#)

[24.13.8.4 \\_CL\\_REF](#)    [24.13.8.5 \\_CL\\_REF1](#)    [24.13.8.6 \\_CL\\_REF2](#)

[24.13.8.7 \\_CL\\_REF3](#)    [24.13.8.8 \\_CL\\_REF4](#)

[24.13.8.9 \\_NULL](#)

[24.13.8.10 \\_AUTO\\_0I, \\_AUTO\\_1J, ... , \\_AUTO\\_9R, \\_AUTO\\_AS, \\_AUTO\\_BT, ... \\_AUTO\\_HZ](#)

[24.13.8.11 \\_AUTO\\_05, \\_AUTO\\_16, \\_AUTO\\_27, ... \\_AUTO\\_49, \\_AUTO\\_AN, \\_AUTO\\_BO, ... \\_AUTO\\_MZ](#)

[24.13.8.12 \\_INS\\_05, \\_INS\\_LR, \\_INS\\_0I](#)

[24.13.8.13 \\_INS\\_SIDE\\_05](#)

[24.13.8.14 \\_SIDE\\_FX](#)

[24.13.8.15 \\_AUTO\\_LAYER\\_LR](#)

[24.13.8.16 \\_APPLY\\_TIN](#)    [24.13.8.17 \\_APPLY\\_DESIGN\\_MODEL](#)    [24.13.8.18 \\_HINGE](#)

[24.13.8.19 \\_SIDE](#)

[24.13.8.20 \\_SILENT\\_NG](#)

[24.13.8.21 \\_PROJECT\\_ATTRIBUTE](#)

### 24.13.8.1 \_AUTO\_LR

Depending on which side the snippet is called from, this parameter will insert the value **L** if it is called from the left side modifiers and **R** if it is called from the right side modifiers.

```
insert "KLIP$_AUTO_LR" "red" 5 unknown -2 $_SCH) 0 $_ECH) 0 absolute
      extra_start extra_end
```

Continue to [24.13.8.2 \\_SCH](#) or return to [24.13.8 Automatic Parameters in Snippets](#) .

### 24.13.8.2 \_SCH

Will insert **named\_position MODIFIER\_START**.

This is typically used to insert the start chainage mode that has been passed to the snippet.

Note that this parameter does **not** include the extension value.

```
insert "KLIP" "red" 5 unknown -2 $(_SCH) 0 $(_ECH) 0 absolute extra_start extra_end
```

Continue to [24.13.8.3 \\_ECH](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.3 \_ECH

Will insert **named\_position MODIFIER\_END**.

This is typically used to insert the end chainage mode that has been passed to the snippet.

Note that this parameter does **not** include the extension value.

```
insert "KLIP" "red" 5 unknown -2 $(_SCH) 0 $(_ECH) 0 absolute extra_start extra_end
```

Continue to [24.13.8.4 \\_CL\\_REF](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.4 \_CL\_REF

Will insert the entire reference string name.

For example, if the reference string name is **RSR01**, \$(\_CL\_REF) will insert **RSR01**

Continue to [24.13.8.5 \\_CL\\_REF1](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.5 \_CL\_REF1

Will insert the last character of the reference string name.

For example, if the reference string name is **RSR01**, \$(\_CL\_REF1) will insert **1**.

Continue to [24.13.8.6 \\_CL\\_REF2](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.6 \_CL\_REF2

Will insert the last two characters of the reference string name.

For example, if the reference string name is **RSR01**, \$(\_CL\_REF2) will insert **01**.

Continue to [24.13.8.7 \\_CL\\_REF3](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.7 \_CL\_REF3

Will insert the last three characters of the reference string name.

For example, if the reference string name is **RSR01**, \$(\_CL\_REF3) will insert **R01**.

Continue to [24.13.8.8 \\_CL\\_REF4](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.8 \_CL\_REF4

Will insert the last four characters of the reference string name.

For example, if the reference string name is **RSR01**, \$(\_CL\_REF4) will insert **SR01**.

Continue to [24.13.8.9 \\_NULL](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.9 \_NULL

The current MTF syntax includes the use of \$null for various values. In snippets, this would be interpreted as a parameter or token null.

Therefore, in snippets, if you would still like to include \$null in MTF commands, they need to be replaced with this automatic parameter.

**An example of MTF modifier syntax** - note the present of \$null values:

```
named_grade "TEST" start_ref 0 final_ref 0 "links" "Design=>CLIN" "Design=>EDGE"
    $null $null $null $null "" absolute extra_start extra_end
```

**The MTF snippet syntax** for it - where \$null has been replaced with \$\_NULL):

```
named_grade "TEST" $_(SCH) 0 $_(ECH) 0 "links" "Design=>CLIN" "Design=>EDGE"
    $_(NULL) $_(NULL) $_(NULL) $_(NULL) "" absolute extra_start extra_end
```

Continue to [24.13.8.10 \\_AUTO\\_0I, \\_AUTO\\_1J, ... , \\_AUTO\\_9R, \\_AUTO\\_AS, \\_AUTO\\_BT, ... \\_AUTO\\_HZ](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.10 \_AUTO\_0I, \_AUTO\_1J, ... , \_AUTO\_9R, \_AUTO\_AS, \_AUTO\_BT, ... \_AUTO\_HZ

This is a set of 18 different, but related, automatic parameters to assist with string naming conventions for multiple occurrences.

These automatic parameters follow the MOSS/MX naming convention for multiple occurrences:

Side	Occurrence																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Left	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
Right	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

**Left side** is 0 to 9, A to H. **Right side** is I to Z.

So, for the 1<sup>st</sup> occurrence of a string, you would use the **\_AUTO\_0I** parameter.

For the 2<sup>nd</sup> occurrence of the string, you would use the **\_AUTO\_1J** parameter, and so on until for the 18<sup>th</sup> occurrence you would use the **\_AUTO\_HZ** parameter.

12d will automatically insert either 0/I, 1/J, 2/K ... H/Z depending on whether the snippet is called on the **left** or **right** side of the modifiers.

Hence the names of the automatic parameter **\_AUTO\_0I** include the **0I** to indicate that it substitutes **0** when on the Left and **I** when on the right.

```
insert "LANE$_(AUTO_0I)" "red" 3.5 unknown -3 $_(SCH) 0 $_(ECH) 0 absolute extra_start extra_end
insert "LANE$_(AUTO_1J)" "red" 3.5 unknown -3 $_(SCH) 0 $_(ECH) 0 absolute extra_start extra_end
insert "LANE$_(AUTO_2K)" "red" 3.5 unknown -3 $_(SCH) 0 $_(ECH) 0 absolute extra_start extra_end
```

Continue to [24.13.8.11 \\_AUTO\\_05, \\_AUTO\\_16, \\_AUTO\\_27, ... \\_AUTO\\_49, \\_AUTO\\_AN, \\_AUTO\\_BO, ... \\_AUTO\\_MZ](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.11 `_AUTO_05`, `_AUTO_16`, `_AUTO_27`, ... `_AUTO_49`, `_AUTO_AN`, `_AUTO_BO`, .... `_AUTO_MZ`

This is a set of 18 different, but related, automatic parameters to assist with string naming conventions for multiple occurrences.

These automatic parameters follow a similar, but alternative naming convention to the MOSS/MX for multiple occurrences:

Side	Occurrence																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Left	0	1	2	3	4	A	B	C	D	E	F	G	H	I	J	K	L	M
Right	5	6	7	8	9	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

**Left side** is 0 to 9, A to H. **Right side** is I to Z.

So, for the 1<sup>st</sup> occurrence of a string, you would use the `_AUTO_05` parameter.

For the 2<sup>nd</sup> occurrence of the string, you would use the `_AUTO_16` parameter, and so on until for the 18<sup>th</sup> occurrence you would use the `_AUTO_MZ` parameter.

12d will automatically insert either 0/5, 1/6, 3/7 ... M/Z depending on whether the snippet is called on the **left** or **right** side of the modifiers.

Hence the names of the automatic parameter `_AUTO_05` include the **05** to indicate that it substitutes **0** when on the Left and **5** when on the right.

```
insert "LANE$_AUTO_05)" "red" 3.5 unknown -3 $_SCH) 0 $_ECH) 0 absolute extra_start extra_end
insert "LANE$_AUTO_16)" "red" 3.5 unknown -3 $_SCH) 0 $_ECH) 0 absolute extra_start extra_end
insert "LANE$_AUTO_27)" "red" 3.5 unknown -3 $_SCH) 0 $_ECH) 0 absolute extra_start extra_end
```

Continue to [24.13.8.12 `\_INS\_05`, `\_INS\_LR`, `\_INS\_0I`](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.12 `_INS_05`, `_INS_LR`, `_INS_0I`

This is used as a normal TEXT parameter, which accepts a single character indicating the side.

Different automatic parameters are used depending on the character expected.

`_INS_05` will insert **0** to **4** for left side, **5** to **9** for right side

`_INS_LR` will insert **L** for left side, **R** for right side

`_INS_0I` will insert **0** to **9** or **A** to **H** for left side, **I** to **Z** for right side

```
// PARAMETER _INS_05 TEXT "Insertion side (0...4 / 5...9)"
// PARAMETER _INS_LR TEXT "Insertion side (L / R)"
// PARAMETER _INS_0I TEXT "Insertion side (0...9,A...H / I...Z)"
```

Once set-up, the `_INS_??` automatic parameter is used to set the `_SIDE_FX` automatic parameter (see [24.13.8.14 `\_SIDE\_FX`](#)).

Note that, as with any other parameter, in order for the `_INS_05` / `_INS_LR` / `_INS_0I` parameter to appear in the MTF Snippet panel, it must be used at least once elsewhere in the snippet. This

can be done in a token definition, directive, or expression.

Continue to [24.13.8.13 \\_INS\\_SIDE\\_05](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.13 \_INS\_SIDE\_05

This is similar to `_INS_05` / `_INS_01` / `_INS_LR`, but without the need to define it as a parameter.

The `_INS_SIDE_05` takes a single character.

The value of `_INS_SIDE_05` is used to determine the value for the `_SIDE_FX` automatic parameter.

`_INS_SIDE_05` will be **0** to **4** for left side, **5** to **9** for right side

The `_INS_SIDE_05` is unique as an automatic parameter, since its value can be set.

In this way, it behaves more like a Directive (See [24.13.11 Snippet Directives](#)).

To set the value of the `_INS_SIDE_05` automatic parameter

```
$_INS_SIDE_05<value>
```

<value> should be a parameter.

The `_INS_SIDE_05` should not be used anywhere else in snippets, including `user_message` commands, except to define its value.

```
// PARAMETER LINK1 TEXT "Attach to link" "EDGE0R"
// Set the _INS_SIDE_05 auto parameter to the 5th character of the LINK1 parameter
$_INS_SIDE_05$(LINK1[5:5])
user_message "_SIDE_FX = $_SIDE_FX" // Display the current value of _SIDE_FX

insert "Design$_SIDE_FX)KLIP0R" "red" 2 unknown -3 $_SCH) 0 $_ECH) 0 absolute
      extra_start  extra_end
```

Continue to [24.13.8.14 \\_SIDE\\_FX](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.14 \_SIDE\_FX

This will insert the link side indicator << for link on left side, or >> for link on right side, depending on the most recent use of `_INS_*`.

That is, if the last state of `_INS_*` corresponds to the left side, `_SIDE_FX` will insert << to indicate insertion to the left.

If the last state of `_INS_*` corresponds to the right side, `_SIDE_FX` will insert >> to indicate insertion to the right. Hence if

`_INS_05` = 0 to 4, then `_SIDE_FX` = <<

`_INS_05` = 5 to 9, then `_SIDE_FX` = >>

`_INS_LR` = L, then `_SIDE_FX` = <<

`_INS_LR` = R, then `_SIDE_FX` = >>

`_INS_01` = 0 to 9, A to H, then `_SIDE_FX` = <<

`_INS_01` = I to Z, then `_SIDE_FX` = >>

`_INS_SIDE_05` = 0 to 4, then `_SIDE_FX` = <<

`_INS_SIDE_05 = 5 to 9, then _SIDE_FX = >>`

Note that in order for the `_SIDE_FX` parameter to be set, the corresponding parameter `_INS_*` must also have been set, given a value (0 to 9, A to Z) and used somewhere in the snippet.

For this reason, if not using the `_INS_05` directly, it is necessary to reference it in a temporary parameter, token or expression.

```
// PARAMETER _INS_05 TEXT "Insertion side (0...4 / 5...9)"
// PARAMETER _INS_LR TEXT "Insertion side (L / R)"
// PARAMETER _INS_OI TEXT "Insertion side (0...9,A...H / I...Z)"
// DISPLAY _INS_05
// DISPLAY _INS_LR
// DISPLAY _INS_OI

// The _INS_* parameter must be used at least once in the snippet
// This can be done in a token, directive, expression or command.
// The $_INS_*) must be used to correctly set the _SIDE_FX
// automatic parameter.
user_message_print "_INS_05 = $_INS_05)// _SIDE_FX now based on this parameter
user_message_print "_SIDE_FX after _INS_05 = $_SIDE_FX) "
insert_absolute "Design$_SIDE_FX)EDGE" "red" 3.5 unknown -3 $_SCH) 0 $_ECH) 0 absolute
    extra_start  extra_end

user_message_print "_INS_LR = $_INS_LR)// _SIDE_FX now based on this parameter
user_message_print "_SIDE_FX after _INS_LR = $_SIDE_FX) "
insert_absolute "Design$_SIDE_FX)EDGE" "yellow" 3.5 unknown -3 $_SCH) 0 $_ECH) 0 absolute
    extra_start  extra_end

user_message_print "_INS_OI = $_INS_OI)// _SIDE_FX now based on this parameter
user_message_print "_SIDE_FX after _INS_OI = $_SIDE_FX) "
insert_absolute "Design$_SIDE_FX)EDGE" "green" 3.5 unknown -3 $_SCH) 0 $_ECH) 0 absolute
    extra_start  extra_end
```

The `_SIDE_FX` allows links to be inserted on any side of the reference, regardless of the side calling the snippet. For example, a snippet on the right side could insert a link on the left side.

Once the `_SIDE_FX` value is used in the snippet, its value will remain the same until re-defined by any of the `_INS_05` / `_INS_LR` / `_INS_OI` / `_INS_SIDE_05` automatic parameters.

Continue to [24.13.8.15 AUTO\\_LAYER\\_LR](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.15 AUTO\_LAYER\_LR

Depending on which side the snippet is called from, will insert the value **layer\_left** if called from the left side modifiers and **layer\_right** if called from the right side modifiers.

This is typically used for the Create String modifier commands.

```
create_strings "$(_AUTO_LAYER_LR)" "Design=>" "*" "$(_SCH) 0 "$(_ECH) 0 "no_colour" ""
"DES ROAD STRS" absolute extra_start extra_end
```

Continue to [24.13.8.16 \\_APPLY\\_TIN](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.16 \_APPLY\_TIN

This will insert the name of the tin from the **Apply MTF** function

```
interface_tin "Design=>INT" "$(_SCH) 0 "$(_ECH) 0 "Design=>" "$(_APPLY_TIN)" 1 2 100 0
"red" "green" 0 0 absolute extra_start extra_end
```

Continue to [24.13.8.17 \\_APPLY\\_DESIGN\\_MODEL](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.17 \_APPLY\_DESIGN\_MODEL

This will insert the name of the **Road Surface Strings** model entered in the **Models** tab of the **Apply MTF** function panel.

```
create_strings "$(_AUTO_LAYER_LR)" "Design=>" "*" "$(_SCH) 0 "$(_ECH) 0 "no_colour" ""
"$(_APPLY_DESIGN_MODEL)" absolute extra_start extra_end
```

Continue to [24.13.8.18 \\_HINGE](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.18 \_HINGE

This will insert the name of the **hinge string**.

Since the user can define a custom hinge name (the default is **HINGE**), this is needed to ensure portability and compatibility with different users and setups.

```
named_grade "CWAY" "$(_SCH) 0 "$(_ECH) 0 "links" "Design=>HING" "Design=>EDGE"
$null $null $null $null "" absolute extra_start extra_end
```

Continue to [24.13.8.19 \\_SIDE](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.19 \_SIDE

Depending on which side the snippet is called from, will insert the value **-1** if called from the left side modifiers and **+1** if called from the right side modifiers.

Continue to [24.13.8.20 \\_SILENT\\_NG](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.20 \_SILENT\_NG

This will insert **\_SILENT\_NAMED\_GRADE**.

When inserted in front of a named grade, the named grade will not appear in any Named Grade choice boxes.

This is typically used for temporary or construction named grades in snippets that you do not wish the end user to see in the GUI.

Continue to [24.13.8.21 PROJECT\\_ATTRIBUTE](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.21 \_PROJECT\_ATTRIBUTE

This will insert a **project attribute** from the current 12d project.

The particular attribute to be inserted is specified between square brackets [ ] and follows the standard convention of attribute paths in **12d Model**.

```
user_message "$(_PROJECT_ATTRIBUTE[ProjectDetails/ProjectNumber/Value])"
```

This will insert the value of the **ProjectNumber** entered in the **Project Details** panel (if ProjectNumber is defined).

Continue to [24.13.8.22 Substrings of parameters](#) or return to [24.13.8 Automatic Parameters in Snippets](#).

### 24.13.8.22 Substrings of parameters

Although there are some preset automatic parameters for extracting a few characters from a string, e.g. **\_CL\_REF3**, it is possible to extract a substring of all **User** parameters, and the **one** special **auto parameter case**, **\_CL\_REF**.

This is done with the following syntax:

```
param[start:end]
```

**param** is the parameter or token name

**start** is the character start position (the first character is at position 1). It must be an integer greater than or equal to 1.

**end** is the character end position (the first character is at position 1). It must be an integer greater than or equal to the start position.

For the value of the selected **param**, only the text from **start** to **end** will be inserted.

Both **start** and **end** are compulsory. The colon is also compulsory.

If **end** is greater than the length of the complete param value, an end position equal to the string length will be used. That is, everything from the start up to the end of the string will be returned.

**start** must be less than **end**.

Substrings are currently supported on all user parameters. Automatic parameters and tokens are not supported.

For example,

if \$(TEXT\_PARAM) has the value **Some text**

then	\$(TEXT_PARAM[1:3])	is	<b>Som</b>
	\$(TEXT_PARAM[4:7])	is	<b>e t</b>

`$(TEXT_PARAM[1:20])` is **Some text**

`$(TEXT_PARAM[5:9])$(TEXT_PARAM[1:4])` is **textSome**

For the special auto parameter case, an example is:

`$( _CL_REF[2:6])`

Continue to the next section [24.13.9 Arithmetic in Snippets](#) or return to [24.13.8 Automatic Parameters in Snippets](#) or [24.13 Defining and Using Snippets](#) .

## 24.13.9 Arithmetic in Snippets

Arithmetic expressions can be used for any of the real values in the lines of a snippet file as long as special rules are obeyed.

1. If a real value is replaced by an expression then the expression must be surrounded by round brackets. That is "(" and ")".
2. If the arithmetic operators plus +, minus -, multiplication \* and division / are used then there must be a space on both sides of them.

For example,  $(\$(TL) - \$(PL))$  and  $(0.0 - (\$(DL) / 2.0) - 0.5)$

Note that the negative of a number has the negative sign hard up against the number with no space in-between. For example **-0.5**

**WARNING** - a future interactive Snippet editor may not be able to support arithmetic in snippets. Those snippets with arithmetic will still need to be modified using a text editor.

An example of some lines from a snippet file using an arithmetic expression is:

```
// RTA Type F Half Kerb to SA transition

// PARAMETER TL REAL "Transition Length" 5.5
// PARAMETER PL REAL "Part Length" 1.340

// insert the links for the 'F' to 'short F' transition

insert "1" "grey" 0.001 0.000 unknown named_position "MODIFIER_START" 0.0 named_position
"MODIFIER_START"  $(\$(TL) - \$(PL))$  absolute extra_start extra_end

insert "2" "grey" 0.499 -0.040 unknown named_position "MODIFIER_START" 0.0 named_position
"MODIFIER_START"  $(\$(TL) - \$(PL))$  absolute extra_start extra_end

...
```

Continue to the next section [24.13.10 Trig and Maths Function Capabilities in Snippets](#) or return to [24.13 Defining and Using Snippets](#).

## 24.13.10 Trig and Maths Function Capabilities in Snippets

There is now some trig and maths capability in the MTF snippet grammar.

The trig functions supported are:

sin(x), cos(x), tan(x) - trig functions

asin(x), acos(x), atan(x) - the inverse trig functions

square(x) - return  $x^2$

sqrt(x)

fabs(x) - returns the absolute value of its x.

ceil(x) - the ceil functions return a floating-point value that represents the smallest integer that is greater than or equal to x.

floor(x) - the floor functions return a floating-point value that represents the largest integer that is less than or equal to x.

log(x) - the log functions return the natural logarithm (base e) of x if successful.

exp(x) - the exp function returns the exponential value of the floating-point parameter, x, if successful. That is, the result is e to the power x, where e is the base of the natural logarithm.

cube(x) - return  $x^3$

pow(x,y) - returns the value of x to the power y.

fmod(x,y) - returns the floating-point remainder of x / y

pi, halfpi, twopi - so you don't have to keep typing the values in

The trig functions etc then allows calculations in a snippet like

```
// Swept angle of 2nd radius.
```

```
//
```

```
@def_tok A2 "(asin(($(D2) - $(R1) + $(DEL_R1_R2) * cos($(A1))) / $(R2)) - $(A1) + halfpi)
```

**Note** - Mathematical formulae in snippets should always be quoted to prevent concatenation and unexpected results.

Continue to the next section [24.13.11.1.1 #define in Snippets - Do Not Use](#) or return to [24.13. Defining and Using Snippets](#).

## 24.13.11 Snippet Directives

See

- [24.13.11.1 Deprecated C Preprocessor from V10 .](#)
- [24.13.11.2 Directives for V11 Onwards .](#)
- [24.13.11.3 Flow control .](#)
- [24.13.11.4 Abbreviations .](#)
- [24.13.11.5 Comparisons of Data Types .](#)
- [24.13.11.6 Tokens and Tokens vs Tokens .](#)
- [24.13.11.7 Tokens vs Strings .](#)
- [24.13.11.8 Tokens vs Doubles .](#)
- [24.13.11.9 Tokens vs Integers .](#)
- [24.13.11.10 Values Defined and Value vs Value .](#)
- [24.13.11.11 Values vs Strings .](#)
- [24.13.11.12 Values vs Doubles .](#)
- [24.13.11.13 Values vs Integers .](#)
- [24.13.11.10 Values Defined and Value vs Value .](#)

Or return to [24.13 Defining and Using Snippets .](#)

### 24.13.11.1 Deprecated C Preprocessor from V10

When Snippets were first introduced in **12d Model 10**, the C preprocessor syntax (e.g. #define or #if / #else / #endif) was used to increase the power of snippets by performing conditional tests and evaluations. The C preprocessor is used extensively throughout **12d Model** already by 12d Programmers (e.g. setup files) but it can be very confusing for users to fully understand.

```
// PARAMETER DL REAL "Drive length" 3.0
// PARAMETER WD REAL "Layback width" 0.6
// PARAMETER DP REAL "Layback depth" -0.04
#define _DR1 (0.0 - ($DL / 2.0) - 0.5)
#define _DR2 (0.0 - ($DL / 2.0))
#define _DR3 ( ($DL / 2.0))
#define _DR4 (($DL / 2.0) + 0.5)
// drop into DW
insert "SAL" "grey" 0.001 0.000 unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
insert "SAI" "grey" 0.499 $DP unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
insert "SAT" "grey" 0.030 0.150 unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
insert "SAB" "grey" 0.180 0.000 unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
```

Due to the complexity and limitations of the existing C preprocessor, a new preprocessor has been created in **12d Model 11** specifically for Snippets.

Snippets in **12d Model 11** can still use the C preprocessor style of directives and snippets written for **12d Model 11** will still continue to run in **12d Model 11**. However, the use of the older style of directives has been superseded (deprecated) in **12d Model 11**.

Continue to [24.13.11.1.1 #define in Snippets - Do Not Use](#) or return to [24.13.11 Snippet Directives](#).

### 24.13.11.1.1 #define in Snippets - Do Not Use

#### IMPORTANT WARNING

From 12d Model 11 onwards, **#define** has been replaced by **@def\_tok** directive and **#define** should not be used.

It is only documented here in case you are working on an old Snippet and need to know what the command does.

**#define** can be used in a snippet file to define parameters for use inside the snippet file.

**WARNING** - a future interactive Snippet editor may not be able to support **#define**. Those snippets with **#define** will still need to be modified using a text editor.

The **#define** is placed at the beginning of a line, followed by one or more spaces and then the `define_name` to be used, and then one or more spaces and the expression that the `define_name` stands for.

```
#define define_name define_expression
```

Then the `define_name` can be used instead of the `define_expression` anywhere else in the snippet file following the **#define**.

**Note** - snippet parameters can be used in the `define_expression` as long as the definition of the parameter occurs before the **#define**.

For example

```
// PARAMETER DL REAL "Drive length" 3.0
// PARAMETER WD REAL "Layback width" 0.6
// PARAMETER DP REAL "Layback depth" -0.04

#define _DR1 (0.0 - ($DL / 2.0) - 0.5)
#define _DR2 (0.0 - ($DL / 2.0) )
#define _DR3 ( ($DL / 2.0) )
#define _DR4 ( ($DL / 2.0) + 0.5)

// drop into DW
insert "SAL" "grey" 0.001 0.000 unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
insert "SAI" "grey" 0.499 $DP unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
insert "SAT" "grey" 0.030 0.150 unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end
insert "SAB" "grey" 0.180 0.000 unknown named_position "MODIFIER_START" _DR1
named_position "MODIFIER_START" _DR2 absolute extra_start extra_end

...
```

Return to [24.13.11.1 Deprecated C Preprocessor from V10.](#)

### 24.13.11.2 Directives for V11 Onwards

The new syntax for directives starts with a **@**, then **a space**, followed by the **directive** and **any additional arguments**.

```
@ <directive> <argument> [<argument>... <argument>]
```

**<directive>** - name of the directive.

**<argument>** - any required arguments for the directive. The number and type of these arguments will depend on the directive.

```
@ if_val_eq "$(KT)" "SM"
// Do something in here
@ end if
```

*For information on all the snippet directives, see the sections*

- [24.13.11.3 Flow control .](#)
- [24.13.11.4 Abbreviations .](#)
- [24.13.11.5 Comparisons of Data Types .](#)
- [24.13.11.6 Tokens and Tokens vs Tokens .](#)
- [24.13.11.7 Tokens vs Strings .](#)
- [24.13.11.8 Tokens vs Doubles .](#)
- [24.13.11.9 Tokens vs Integers .](#)
- [24.13.11.10 Values Defined and Value vs Value .](#)
- [24.13.11.11 Values vs Strings .](#)
- [24.13.11.12 Values vs Doubles .](#)
- [24.13.11.13 Values vs Integers .](#)
- [24.13.11.10 Values Defined and Value vs Value .](#)

Or return to [24.13.11 Snippet Directives](#) or [24.13 Defining and Using Snippets](#) .

### 24.13.11.3 Flow control

In a snippet, the normal processing flow is that each snippet command is processed in the order in which they are encountered. It is possible, however, to change the order in which such commands are processed or even to totally skip commands. These are done through **flow control directives** in the snippets.

See

[24.13.11.3.1 Conditional flow control.](#)

[24.13.11.3.2 Transferring Flow Control.](#)

Or return to [24.13.11 Snippet Directives](#) or [24.13 Defining and Using Snippets](#).

### 24.13.11.3.1 Conditional flow control

Snippet commands can be processed or ignored based on the result of a test.

This is known as a conditional test.

Snippet directives based on conditional statements (*i.e.* testing whether a statement is true or false) are generally of the form **if/else/end if**.

```
@ if...
@ end_if
```

OR

```
@ if...
@ else
@ end_if
```

When the **if** condition is **true**, commands will be processed until the **else** or **end\_if** directive is found.

When the **if** condition is **false**, commands will be ignored **until** the **else** or **end\_if** directive is found.

When the **if** condition is **false** and an **else** directive is found the commands following the **else** until the **end\_if** will be processed.

An **end\_if** directive must always be present.

An **else** directive cannot exist without a preceding **if** directive.

There is **no else\_if** directive.

```
@ if...
// When the if condition is true, the commands in this section,
// until the else or end_if will be processed.
@ else
// When the if condition is false, the commands in this section,
// until the end_if will be processed
@ end_if
```

**Embedding if** directives within other conditional directives (if or else), *i.e.* nested ifs, are **not allowed**.

Continue to [24.13.11.3.2 Transferring Flow Control](#) or return to [24.13.11.3 Flow control](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.3.2 Transferring Flow Control

In addition to the conditional flow control, it is possible within snippets to transfer processing to an entirely different and disconnected section of the snippet.

These transfers (jumps) are performed by the **goto** and **label** directives.

See

[24.13.11.3.2.1 Goto - transfer processing to a given location .](#)

[24.13.11.3.2.2 Label - mark a location for goto](#)

Or return to [24.13 Defining and Using Snippets .](#)

#### 24.13.11.3.2.1 Goto - transfer processing to a given location

The **goto** directive transfers processing to a corresponding **label** directive within the snippet.

The syntax is:

```
@ goto <label>
```

**<label>** - is the name of the label to which control will be transferred. One and only one label should be specified.

The **goto** directive must be defined with a corresponding **label** directive.

**Control** can only be transferred **forwards** in a file (i.e. further down the contents of the file); control **cannot** be passed **backwards**.

The corresponding **label** should **not** be inside a conditional flow group, if/else/end\_if.

```
// PARAMETER KT TEXT "Kerb type" OPTIONAL

@ def_tok EXTRAS "absolute extra_start extra_end"

@ if_val_neq "$ (KT)" "SA Kerb"
  @ goto fred
  user_message "Should not get here." $_SCH) 0 $_ECH) 0 $(EXTRAS)
@ end_if

user_message "Should not get here." $_SCH) 0 $_ECH) 0 $(EXTRAS)

@ label fred

user_message "Should get here." $_SCH) 0 $_ECH) 0 $(EXTRAS)
```

Continue to [24.13.11.3.2.2 Label - mark a location for goto](#) or return to [24.13.11.3.1 Conditional flow control](#) or [24.13.11.3 Flow control](#) or [24.13.11 Snippet Directives .](#)

### 24.13.11.3.2.2 Label - mark a location for goto

The **label** directive is used in conjunction with a **goto** directive to indicate a position within the snippet.

The syntax is:

```
@ label <label>
```

**<label>** - is the name of this label. A label should normally be paired with a corresponding **goto** directive, though it is not compulsory. A label can exist without a **goto**.

The **label** must be placed **after** the **goto** directive to which it corresponds. Control can only be transferred forwards in a file (i.e. further down the contents of the file); control cannot be passed backwards.

The **label** directive should not be placed inside a conditional flow group, `if/else/end_if`.

The label does not have to be unique within the file and multiple pairs of `goto/labels` are permitted. Note, however, that `goto` will simply jump to the first/next found matching label.

```
@ goto end_earthwrks  
@ label end_earthwrks  
.  
@ goto end_earthwrks  
@ label end_earthwrks
```

Continue to [24.13.11.4 Abbreviations](#) or return to [24.13.11.3.1 Conditional flow control](#) or [24.13.11.3 Flow control](#) or [24.13.11 Snippet Directives](#) .

#### 24.13.11.4 Abbreviations

When reading or writing snippet directives, it is useful to know and remember the abbreviations below.

**if** - conditional test

**def** - define a token

**tok** - token

**val** - value

**str** - string (i.e. text)

**int** - integer

**dbl** - double

**tol** - tolerance

**eq** - equal

**neq** - not equal

**gt** - greater than

**lt** - less than

**ge** - greater than or equal to

**le** - less than or equal to

**in\_range** - within a range of values defined by lower and upper bounds

**nin\_range** - not within (i.e. outside) a range of values defined by lower and upper bounds

These abbreviations can be used to determine the appropriate directive to use or interpret and understand directives in use.

For example, the **if\_val\_le\_dbl\_tol**, is a conditional directive (**if**) to test if a value (**val**) is less than or equal to (**le**) any of double values (**dbl**) with a user-specified tolerance (**tol**).

Continue to [24.13.11.5 Comparisons of Data Types](#) or return to [24.13.11 Snippet Directives](#).

### 24.13.11.5 Comparisons of Data Types

There are two basic types of comparison used in snippets.

- (a) The **first** is **character-based** and is typically used on strings of text.

In a character-based comparison of two values, the first character from one string is compared against the first character from the other string. If both characters are equal, processing continues to the next character in each string.

Character comparisons in snippets are **not case sensitive**.

```
@ def_tok TOK1 "SOME TEXT"

@ if_tok_eq_str TOK1 "some text"
  user_message "This will be true because comparisons are case insensitive"
@ end_if
```

Character comparisons should not be used to compare two numbers. All directives including the **str** abbreviation, e.g. `if_tok_eq_str`, `if_val_neq_str`, use character-based comparisons.

```
@ def_tok TOK1 1

@ if_tok_eq_str TOK1 1.0
@ else
  user_message "False because if_tok_eq_str uses a character-based comparison"
@ end_if
```

- (b) The **second** type of comparison is **numeric** and is typically used on numbers both integers and doubles.

In a numerical-based comparison of two values, each value is first converted into a double. If the value cannot be converted to a number, then an error is typically generated.

```
@ def_tok TOK1 1 // This is a valid number
@ def_tok TOK2 2.345 // As is this
@ def_tok TOK3 "1.78" // This can still be converted to a valid number
@ def_tok TOK4 "1e-4" // As can this
@ def_tok TOK5 "RL1.234" // But not this

@ if_tok_eq_dbl TOK5 "1.234"
  user_message "TOK5 is not a valid number"
@ else
  user_message "TOK5 cannot be converted to a valid number"
@ end_if
```

Assuming both values in a numerical comparison are valid numbers (or can be converted as such), the numbers are then compared in standard mathematical fashion.

```
@ def_tok TOK1 1

@ if_tok_eq_dbl TOK1 1.0
  user_message "True because if_tok_eq_dbl uses a numerical-based comparison"
@ end_if

@ if_tok_eq_dbl TOK1 "0.001e3"
  user_message "Also true because if_tok_eq_dbl uses a numerical-based comparison"
@ end_if
```

Both double and integer numbers can be used in directive comparisons for doubles (contain **dbl** abbreviation). However, only integer numbers can be used in directive comparisons for integers (contain **int** abbreviation). Doubles will not be converted or truncated to integers in such cases, but instead produce an error.

Continue to [24.13.11.6 Tokens and Tokens vs Tokens](#) or return to [24.13.11 Snippet Directives](#).

### 24.13.11.6 Tokens and Tokens vs Tokens

The following directives define tokens, check whether tokens are defined and compare tokens with other tokens.

See

[24.13.11.6.1 def\\_tok](#)

[24.13.11.6.2 if\\_tok](#)

[24.13.11.6.3 if\\_ntok](#)

[24.13.11.6.4 if\\_tok\\_ntok](#)

[24.13.11.6.5 if\\_two\\_toks](#)

[24.13.11.6.6 if\\_two\\_toks\\_eq](#)

[24.13.11.6.7 if\\_all\\_toks](#)

[24.13.11.6.8 if\\_nall\\_toks](#)

[24.13.11.6.9 if\\_tok\\_eq\\_tok](#)

[24.13.11.6.10 if\\_tok\\_neq\\_tok](#)

[24.13.11.6.11 def\\_tok\\_minus](#)

[24.13.11.6.12 def\\_tok\\_minus\\_inc](#)

Or return to [24.13 Defining and Using Snippets](#) .

### 24.13.11.6.1 def\_tok

The **def\_tok** directive **defines** a **token** that can have an **optional value**.

The syntax is:

```
@ def_tok <token> [<string>... <string>]
```

**<token>** - this is the name of the token.

Wherever the token is to be used in the snippet, put **\$(token)**

Note that in the older V10 syntax, this was \$token.

The characters of the token name can only be alphanumeric (upper and lower case) and underscores but the name **cannot contain spaces**.

The **token names** are **not case sensitive**.

so two names differing only by case are considered to be identical. *i.e.* THIS, This, this and tHis are all considered identical by 12d Model.

Tokens are treated similarly to parameters, so any name must be unique within a snippet for all parameters and tokens.

**<string>** - one or more optional strings of character (text).

Any string with spaces must be enclosed in quotes.

Multiple strings are concatenated.

The string can contain other tokens, though any tokens must have previously been defined.

The string can also contain mathematical expressions.

The snippet preprocessor will preserve any white space within quoted strings.

However, multiple strings or values will be concatenated without any white space.

If a string has spaces that need to be retained, enclose the string in quotes.

Tokens are used and substituted in the same way as parameters, *e.g.* \$(TOKEN).

```
@ def_tok TOK //defines a token TOK with no value
@ def_tok TOK A //defines a token TOK with the value "A"
@ def_tok TOK "A B" //defines a token TOK with the value "A B"
@ def_tok TOK A B //defines a token TOK with the value "AB", concatenates
@ def_tok TOK A B "C D" //defines a token TOK with the value "ABC D", concatenates
@ def_tok TOK $(KT) //defines a token TOK with the value of the substituted parameter
```

Continue to [24.13.11.6.2 if\\_tok](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.2 if\_tok

The **if\_tok** directive tests whether **any** of the given **tokens** are **defined**.

The syntax is:

```
@ if_tok <token> [<token>... <token>]
// Commands to be processed when statement is true
@ end_if
```

**<token>** - the name of the token to test.

At least one token must be specified, however, multiple tokens can be included.

Multiple tokens should be separated by whitespace.

If any of the tokens are defined, the statement is true.

If none of the tokens are defined, the statement is false.

Multiple tokens are handled as if joined by logical OR statements.

**NOTE:** This directive simply tests whether the token is defined and not whether the value is valid or blank.

```
@ def_tok "T1"
@ def_tok "T2"

@ if_tok "T1"
  user_message "Found token T1"
@ end_if

@ if_tok "T1" "T2"
  user_message "Found token T1 or T2"
@ end_if

@ if_tok "T2"
  user_message "Found token T2"
@ end_if
```

See also:

[24.13.11.6.3 if\\_ntok](#) - for logically opposite directive

[24.13.11.6.7 if\\_all\\_toks](#) - for testing if all tokens are defined

Continue to [24.13.11.6.3 if\\_ntok](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.3 if\_ntok

The **if\_ntok** directive tests whether **particular tokens** are **undefined**.

The syntax is:

```
@ if_ntok <token> [<token>... <token>]
// Commands to be processed when statement is true
@ end_if
```

**<token>** - the name of the token to test.

At least one token must be specified, however, multiple tokens can be included.

Multiple tokens should be separated by whitespace.

If **none** of the **tokens** are **defined**, the statement is **true** and the snippet commands until the next end\_if directive are processed.

If **any** of the **tokens** are **defined**, the statement is **false** and the entire block of commands (until the next end\_if directive) are skipped.

**Multiple tokens** are handled as if **joined** by **logical OR** statements.

**NOTE:** This directive simply tests whether the token is defined and not whether the value is valid or blank.

```
@ def_tok T1
@ def_tok T2

@ if_ntok T1
  user_message "Didn't find token T1"
@ end_if

@ if_ntok T1 T2
  user_message "Didn't find token T1 or T2"
@ end_if

@ if_ntok T2
  user_message "Didn't find token T2"
@ end_if
```

See also:

[24.13.11.6.2 if\\_tok](#) - logically opposite directive

[24.13.11.6.8 if\\_nall\\_toks](#) - for testing if all tokens are undefined

Continue to [24.13.11.6.4 if\\_tok\\_ntok](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

#### 24.13.11.6.4 if\_tok\_ntok

The **if\_tok\_ntok** directive tests if **any** token in **one set** is **defined** and if **any** token in **another set** is **not defined**.

The syntax is:

```
@ if_tok_ntok <token> [<token>... <token>] ; <token> [<token>... <token>]
```

**<token>** - the name of the token to test.

At least one token in each set must be specified, however, multiple tokens can be included.

Multiple tokens should be separated by whitespace.

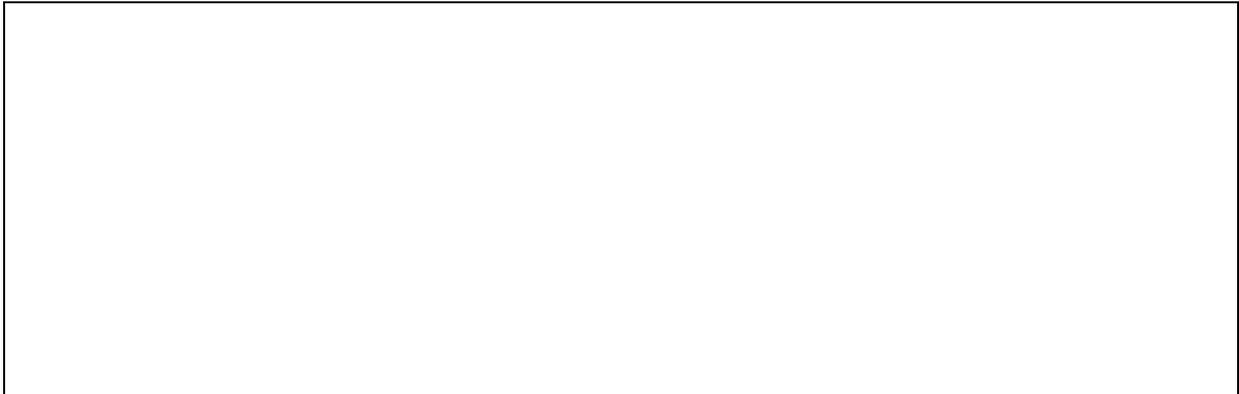
Two sets of tokens are separated by a semi-colon, which must be included.

Tokens in the first set, between the **if\_tok\_ntok** and the semi-colon, are tested to see if they are defined. Tokens in the second set, between the semi-colon and end, are tested to see if they are not defined.

If any of the tokens in the first set are defined and any of the tokens in the second set are undefined, the statement is true.

If none of the tokens in the first set are defined or if all of the tokens in the second set are defined, the statement is false.

Note that both sets must be true for the overall statement to be true. If either set is false, then the overall statement will be false.



```
@ def_tok A
@ def_tok B
// token C is undefined
@ if_tok_ntok A ; C
  user_message "Token A or B are defined, but token C is undefined"
@ end_if

@ if_tok_ntok A ; B
  user_message "True: Token A is defined and Token B is undefined"
@ else
  user_message "False: Token A is defined, but token B is not undefined"
@ end_if

@ if_tok_ntok A B ; C
  user_message "True: Token A or B is defined and Token C is undefined"
@ end_if

@ if_tok_ntok C ; A
  // Won't be true in this example
@ else
  user_message "False: Token C is not defined or Token A is not undefined"
@ end_if
```

Continue to [24.13.11.6.5 if\\_two\\_toks](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.5 if\_two\_toks

The **if\_two\_toks** directive tests the result of a logical comparison of if two tokens are defined.

The syntax is:

```
@ if_two_toks <token1> <op> <token2>
// Commands to be processed when statement is true
@ end_if
```

**<token1>** - the name of the token to test. Must be only one token

**<op>** - the logical operator to use for the two tokens. This can be **&&** for logical AND or **||** for logical OR.

**<token2>** - the name of the token to test. Must be only one token.

The conditional result of this directive is dependent on the two tokens and the logical operator. The combination of possible values and their results are in the table below.

token1	op	token2	Result
defined	&&	defined	true
defined	&&	undefined	false
undefined	&&	defined	false
undefined	&&	undefined	false
defined		defined	true
defined		undefined	true
undefined		defined	true
undefined		undefined	false

```
@ def_tok T1
@ def_tok T2
// Token T3 is not defined

@ if_two_toks T1 && T2
  user_message "Tokens T1 and T2 are both defined"
@ end_if

@ if_two_toks T1 || T2
  user_message "Token T1 or T2 is defined"
@ end_if

@ if_two_toks T1 && T3
  user_message "Token T1 and T3 are both defined" // This shouldn't be shown
@ else
  user_message "Token T1 and T3 are not both defined" // This should be shown
@ end_if

@ if_two_toks T1 || T3
  user_message "Token T1 or T3 is defined"
@ end_if
```

See also:

[24.13.11.6.6 if\\_two\\_toks\\_eq](#) - for logical comparison against values

Continue to [24.13.11.6.6 if\\_two\\_toks\\_eq](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.6 if\_two\_toks\_eq

The **if\_two\_toks\_eq** directive tests the result of a two tests of tokens against corresponding values.

The syntax is:

```
@ if_two_toks_eq <token1> <value1> <op> <token2> <value2>
// Commands to be processed when statement is true
@ end_if
```

**<token1>** - the name of the first token to test. Must be only one token.

**<value1>** - the value to compare against token1.

**<op>** - the logical operator to use for the two tokens.

This can be && for logical AND or || for logical OR

**<token2>** - the name of the second token to test. Must be only one token.

**<value2>** - the value to compare against token2.

When evaluating the statement, token1 is compared against value1 and token2 is compared against value2. The results of both comparisons are then combined using the logical operator, **op**, and evaluated to set the final result of the if\_two\_toks\_eq directive.

token1	value1	token1 = value1	op	token2 = value2	token2	value2	Overall Result
A	A	true	&&	true	B	B	true
A	C	false	&&	true	B	B	false
A	A	true	&&	false	B	D	false
A	C	false	&&	false	B	D	false
A	A	true		true	B	B	true
A	C	false		true	B	B	true
A	A	true		false	B	D	true
A	C	false		false	B	D	false

```
@ def_tok TOK_A "alpha"
@ def_tok TOK_B "bravo"

@ if_two_toks_eq TOK_A "alpha" || TOK_B "bravo"
  user_message "TOK_A = alpha or TOK_B = bravo"
@ end_if

@ if_two_toks_eq TOK_A "charlie" && TOK_B "bravo"
  user_message "TOK_A = charlie and TOK_B = bravo" // This should not be output
@ else
  user_message "TOK_A != charlie or TOK_B != bravo" // This should be output
@ end_if
```

See also:

[24.13.11.6.5 if\\_two\\_toks](#) - testing if two tokens are defined with logical operator

Continue to [24.13.11.6.7 if\\_all\\_toks](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.7 if\_all\_toks

The **if\_all\_toks** directive tests whether all given tokens are defined.

The syntax is:

```
@ if_all_toks <token> [<token>... <token>]
// Commands to be processed when statement is true
@ end_if
```

**<token>** - the name of the token to test.

At least one token must be specified, however, multiple tokens can be included.

Multiple tokens should be separated by whitespace.

If all of the tokens are defined, the statement is true.

If any of the tokens are undefined, the statement is false.

Multiple tokens are handled as if joined by logical AND statements.

**NOTE:** This directive simply tests whether the token is defined and not whether the value is valid or blank.

```
@ def_tok T1
@ def_tok T2
@ def_tok T3

@ if_all_toks T1
  user_message "Found token T1"
@ end_if

@ if_all_toks T1 T2
  user_message "Found token T1 and T2"
@ end_if

@ if_all_toks T1 T2 T3
  user_message "Found token T1, T2 and T3"
@ end_if
```

See also:

[24.13.11.6.8 if\\_nall\\_toks](#) - logically opposite directive

[24.13.11.6.2 if\\_tok](#) - test if any given tokens are defined

Continue to [24.13.11.6.8 if\\_nall\\_toks](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.8 if\_nall\_toks

The **if\_nall\_toks** directive tests whether all of the given tokens are undefined.

The syntax is:

```
@ if_nall_toks <token> [<token>... <token>]
// Commands to be processed when statement is true
@ end_if
```

**<token>** - the name of the token to test.

At least one token must be specified, however, multiple tokens can be included. Multiple tokens should be separated by whitespace.

If all of the tokens are undefined, the statement is true.

If any of the tokens are defined, the statement is false.

Multiple tokens are handled as if joined by logical AND statements.

**NOTE:** This directive simply tests whether the token is defined and not whether the value is valid or blank.

```
@ def_tok T1
@ def_tok T2
@ def_tok T3

@ if_nall_toks T1
  user_message "Didn't find token T1"
@ end_if

@ if_nall_toks T1 T2
  user_message "Didn't find tokens T1 or T2"
@ end_if

@ if_nall_toks T1 T2 T3
  user_message "Didn't find tokens T1, T2 or T3"
@ end_if
```

See also:

[24.13.11.6.7 if\\_all\\_toks](#) - logically opposite directive

[24.13.11.6.3 if\\_ntok](#) - testing if any given tokens are undefined

Continue to [24.13.11.6.9 if\\_tok\\_eq\\_tok](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.9 if\_tok\_eq\_tok

The **if\_tok\_eq\_tok** directive tests whether a token is equal to another token.

The syntax is:

```
@ if_tok_eq_tok <token1> <token2>
// Commands to be processed when statement is true
@ end_if
```

**<token1>** - the first token to be compared.

**<token2>** - the second token to be compared.

**token1** and **token2** are compared using a character comparison (*i.e.* text-based, not numeric).

If the token1 and token2 values are equal, the statement is true.

If the token1 and token2 values do not match, the statement is false.

If either or both token1 or token2 are undefined, the statement is false.

```
@ def_tok TOK1 "alpha"
@ def_tok TOK2 "bravo"

@ if_tok_eq TOK1 TOK2
  user_message "Tokens 1 and 2 are equal"
@ else
  user_message "Tokens 1 and 2 are not equal or undefined"
@ end_if
```

See also:

[24.13.11.6.10 if\\_tok\\_neq\\_tok](#) - logically opposite directive

if\_tok\_eq\_str - testing against a string

if\_tok\_eq\_dbl - testing against a double

if\_tok\_eq\_int - testing against an integer

Continue to [24.13.11.6.10 if\\_tok\\_neq\\_tok](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.6.10 if\_tok\_neq\_tok

The **if\_tok\_neq\_tok** directive tests whether a token is not equal to another token.

The syntax is:

```
@ if_tok_neq_tok <token1> <token2>
// Commands to be processed when statement is true
@ end_if
```

**<token1>** - the first token to be compared.

**<token2>** - the second token to be compared.

**Token1** and **token2** are compared using a character comparison (i.e. text-based, not numeric).

If token1 and token2 values do not match, the statement is true.

If token1 and token2 values match, the statement is false.

If either token1 or token2 is undefined, an error message is produced and the statement is evaluated as true.

```
@ def_tok TOK1 "alpha"
@ def_tok TOK2 "bravo"

@ if_tok_neq_tok TOK1 TOK2
  user_message "Tokens 1 and 2 are not equal"
@ else
  user_message "Tokens 1 and 2 are equal"
@ end_if
```

See also:

[24.13.11.6.9 if\\_tok\\_eq\\_tok](#) - logically opposite directive

[if\\_tok\\_neq\\_str](#) - comparison against string

[if\\_tok\\_neq\\_dbl](#) - comparison against double

[if\\_tok\\_neq\\_int](#) - comparison against integer

Continue to [24.13.11.6.11 def\\_tok\\_minus](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.11 def\_tok\_minus

The **def\_tok\_minus** directive defines a token as the result of a single digit or character subtracted from another.

The syntax is:

`@ def_tok_minus <token> <value1> <value2>`

**<token>** - the name of the token to define.

**<value1>** - a **single** integer or character or parameter that evaluates to such

**<value2>** - a **single** integer or character or parameter that evaluates to such

The value of token is taken as the result of subtracting value2 from value1.

To determine the value of <token>, **value1** and **value2** are converted to their equivalent numerical ASCII character code (e.g. 0 = 48, 9 = 57, A = 65, Z = 90, a = 97, z = 122).

The ASCII character code for value2 is subtracted from the ASCII character code for value1. The result is set as the value of <token>.

**Example with integers:**

Raw snippet values:     <value1> = 5, <value2> = 2

Equivalent ASCII character codes:<value1> = 53, <value2> = 50

Token value = 53 - 50 = 3

**Example with letters:**

Raw snippet values:     <value1> = D, <value2> = B

Equivalent ASCII character codes:<value1> = 68, <value2> = 66

Token value = 68 - 66 = 2

For single-digit integers, this is the same as the basic arithmetic on the integers.

For this reason, it is recommended that <value1> and <value2> are of the same type and case. That is, both values are:

- *integers (0-9)*
- *upper-case single letters (A-Z); or*
- *lower-case single letters (a-z).*

Mixing value types - e.g. integer and letter - will likely result in unexpected behaviour.

It is possible for <value1> or <value2> to contain parameters, e.g. "\$ (WIDTH)", so long as the parameters evaluate to a single digit or character.

It is not possible to use tokens in either <value1> or <value2> due to the order in which snippets are processed and tokens substituted. For more information see [24.13.13 Order of snippet processing](#).

If value1 or value2 is not a single character, an error message will be produced.

```
// PARAMETER LANE1 INTEGER "Start lane" 5
// PARAMETER LANE2 INTEGER "End lane" 2

// ASCII character codes:
// 5 = 53
// 2 = 50

@ def_tok_minus NUM_LANES $(LANE1) $(LANE2)
// For LANE1 = 5, LANE2 = C, NUM_LANES = 53 - 50 = 3

user_message "Number of lanes between start and end = $(NUM_LANES)"
```

```
// PARAMETER LANE1 TEXT "Start lane character (A-Z)" "F"
// PARAMETER LANE2 TEXT "End lane character (A-Z)" "C"

// ASCII character codes:
// F = 70
// C = 67

@ def_tok_minus NUM_LANES $(LANE1) $(LANE2)
// For LANE1 = F and LANE2 = C, NUM_LANES = 70 - 67 = 3

user_message "Number of lanes between start and end = $(NUM_LANES)"
```

**TIP** - To ensure value1 and value2 are single characters, use the substring functionality to only use the first character.

```
// PARAMETER LANE1 TEXT "Start lane character (A-Z)" "FAST"
// PARAMETER LANE2 TEXT "End lane character (A-Z)" "CAR"

// Using substring functionality to only use the 1st character of each parameter
@ def_tok_minus NUM_LANES $(LANE1[1:1]) $(LANE2[1:1])

user_message "Number of lanes between start and end = $(NUM_LANES)"
```

Continue to [24.13.11.6.12 def\\_tok\\_minus\\_inc](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.6.12 def\_tok\_minus\_inc

The **def\_tok\_minus\_inc** defines a token with the result of a single digit or single-character subtracted from another, plus one.

That is, it is similar to **def\_tok\_minus**, but the result is one more than the equivalent **def\_tok\_minus** value (i.e. it is an inclusive subtraction).

The syntax is:

@ def\_tok\_minus\_inc <token> <value1> <value2>

**<token>** - the name of the token to define.

**<value1>** - a single integer or character or parameter that evaluates to such

**<value2>** - a single integer or character or parameter that evaluates to such

The value of token is taken as the result of subtracting value2 from value1 then adding one.

To determine the value of <token>, value1 and value2 are converted to their equivalent numerical ASCII character code (e.g. 0 = 48, 9 = 57, A = 65, Z = 90, a = 97, z = 122).

The ASCII character code for value2 is subtracted from the ASCII character code for value1 and one (1) is added. The result is set as the value of <token>.

**Example with integers:**

Raw snippet values:      <value1> = 5, <value2> = 2

Equivalent ASCII character codes:<value1> = 53, <value2> = 50

Token value = 53 - 50 + 1 = 4

**Example with letters:**

Raw snippet values:      <value1> = D, <value2> = B

Equivalent ASCII character codes:<value1> = 68, <value2> = 66

Token value = 68 - 66 + 1 = 3

For single-character integers, this is the same as the basic arithmetic on the integers.

For this reason, it is recommended that <value1> and <value2> are of the same type and case. That is, both values are:

- *integers (0-9)*
- *upper-case single letters (A-Z); or*
- *lower-case single letters (a-z).*

Mixing value types - e.g. integer and letter - will likely result in unexpected behaviour.

It is possible for <value1> or <value2> to contain parameters, e.g. "\$ (WIDTH)", so long as the parameters evaluate to a single character.

It is not possible to use tokens in either <value1> or <value2> due to the order in which snippets are processed and tokens substituted. For more information see [24.13.13 Order of snippet processing](#).

If value1 or value2 is not a single character, an error message will be produced.

```
// PARAMETER LANE1 INTEGER "Start lane" 5
// PARAMETER LANE2 INTEGER "End lane" 2

// ASCII character codes:
// 5 = 53
// 2 = 50

@ def_tok_minus_inc NUM_LANES $(LANE1) $(LANE2)
// For LANE1 = 5, LANE2 = C, NUM_LANES = 53 - 50 + 1 = 4

user_message "Number of lanes between start and end (inclusive) = $(NUM_LANES)"
```

```
// PARAMETER LANE1 TEXT "Start lane character (A-Z)" "F"
// PARAMETER LANE2 TEXT "End lane character (A-Z)" "C"

// ASCII character codes:
// F = 70
// C = 67

@ def_tok_minus NUM_LANES $(LANE1) $(LANE2)
// For LANE1 = F and LANE2 = C, NUM_LANES = 70 - 67 + 1 = 4

user_message "Number of lanes between start and end (inclusive) = $(NUM_LANES)"
```

**TIP** - To ensure value1 and value2 are single characters, use the substring functionality to only use the first character.

```
// PARAMETER LANE1 TEXT "Start lane character (A-Z)" "FAST"
// PARAMETER LANE2 TEXT "End lane character (A-Z)" "CAR"

// Using substring functionality to only use the 1st character of each parameter
@ def_tok_minus NUM_LANES $(LANE1[1:1]) $(LANE2[1:1])

user_message "Number of lanes between start and end (inclusive) = $(NUM_LANES)"
```

Continue to [24.13.11.7 Tokens vs Strings](#) or return to [24.13.11.6 Tokens and Tokens vs Tokens](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.7 Tokens vs Strings

The following directives compare a token value against given strings.

See

[24.13.11.7.1 if\\_tok\\_eq\\_str](#)

[24.13.11.7.2 if\\_tok\\_neq\\_str](#)

#### 24.13.11.7.1 if\_tok\_eq\_str

The **if\_tok\_eq\_str** directive tests if a token's **value** is **equal** to **any** given **strings**.

The syntax is:

```
@ if_tok_eq_str <token> <string> [<string>... <string>]
// Commands to be processed if statement is true
@ end_if
```

**<token>** - the name of the token to compare.

**<string>** - the string to compare against the value of token.

At least one string must be specified; however, multiple strings can be included.

Multiple strings should be separated by whitespace. Strings should be enclosed in quotes.

If the **value** of **token** is **equal** to **any** of the **strings**, the statement is **true**.

If the value of token is not equal to any of the strings (i.e. the token value is equal to none of the given strings), the statement is false.

Multiple strings are handled as if joined by logical OR statements.

```
@ def_tok A_TOKEN "a token"
@ def_tok B_TOKEN "b token"

@ if_tok_eq_str A_TOKEN "a token" "b token" // must equal one of the list, should pass
  user_message "if_tok_eq_str = TRUE , A = a token OR b token"

@ if_tok_eq_str A_TOKEN "x token" "y token" // must equal one of the list, should fail
@ else
  user_message "if_tok_eq_str = FALSE , A != x token OR y token"
@ end_if
```

Continue to [24.13.11.7.2 if\\_tok\\_neq\\_str](#) or return to [24.13.11.7 Tokens vs Strings](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.7.2 if\_tok\_neq\_str

The **if\_tok\_neq\_str** directive tests if a token is **not equal** to **any** of the given **strings**.

The syntax is:

```
@ if_tok_neq_str <token> <string> [<string>... <string>]
// Commands to be processed if statement is true
@ end_if
```

**<token>** - the name of the token to compare.

**<string>** - the string to compare against the value of token.

At least one string must be specified, however, multiple strings can be included. Multiple strings should be separated by whitespace. Strings should be enclosed in quotes.

If the value of token is not equal to any of the strings, the statement is true.

If the value of token is equal to any of the string values, the statement is false.

Multiple strings are handled as if joined by logical OR statements.

```
@ def_tok A_TOKEN "a token"
@ def_tok B_TOKEN "b token"

@ if_tok_eq_str A_TOKEN "a token" "b token" // must equal one of the list, should pass
  user_message "if_tok_eq_str = TRUE , A = a token OR b token"

@ if_tok_eq_str A_TOKEN "x token" "y token" // must equal one of the list, should fail
@ else
  user_message "if_tok_eq_str = FALSE , A != x token OR y token"
@ end_if
```

Continue to [24.13.11.8 Tokens vs Doubles](#) or return to [24.13.11.7 Tokens vs Strings](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8 Tokens vs Doubles

The following directives compare a token value against given doubles.

In these directives, **<double>** can be either a real number or an evaluation of a previously defined token. For example \$(T49).

See

- [24.13.11.8.1 if\\_tok\\_eq\\_dbl](#)
- [24.13.11.8.2 if\\_tok\\_eq\\_dbl\\_tol](#)
- [24.13.11.8.3 if\\_tok\\_neq\\_dbl](#)
- [24.13.11.8.4 if\\_tok\\_neq\\_dbl\\_tol](#)
- [24.13.11.8.5 if\\_tok\\_lt\\_dbl](#)
- [24.13.11.8.6 if\\_tok\\_lt\\_dbl\\_tol](#)
- [24.13.11.8.7 if\\_tok\\_le\\_dbl](#)
- [24.13.11.8.8 if\\_tok\\_le\\_dbl\\_tol](#)
- [24.13.11.8.9 if\\_tok\\_gt\\_dbl](#)
- [24.13.11.8.10 if\\_tok\\_gt\\_dbl\\_tol](#)
- [24.13.11.8.11 if\\_tok\\_ge\\_dbl](#)
- [24.13.11.8.12 if\\_tok\\_le\\_dbl\\_tol](#)
- [24.13.11.8.13 if\\_tok\\_gt\\_dbl](#)
- [24.13.11.8.14 if\\_tok\\_gt\\_dbl\\_tol](#)
- [24.13.11.8.15 if\\_tok\\_ge\\_dbl](#)
- [24.13.11.8.16 if\\_tok\\_ge\\_dbl\\_tol](#)
- [24.13.11.8.17 if\\_tok\\_in\\_range\\_dbl](#)
- [24.13.11.8.18 if\\_tok\\_in\\_range\\_dbl\\_tol](#)
- [24.13.11.8.19 if\\_tok\\_nin\\_range\\_dbl](#)
- [24.13.11.8.20 if\\_tok\\_nin\\_range\\_dbl\\_tol](#)

Or return to [24.13 Defining and Using Snippets](#).

### 24.13.11.8.1 if\_tok\_eq\_dbl

The **if\_tok\_eq\_dbl** directive tests if a token's **value** is **equal** to **any** given **double** values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (i.e. 1.0E-6).

The syntax is:

```
@ if_tok_eq_dbl <token> <double> [<double>... <double>]
// Commands processed if statement is true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token.

**double** can be a real number or the evaluation of a token eg \$(T49).

At least one double value must be specified, however, multiple double values can be included.

Multiple double values should be separated by whitespace.

If the value of token is equal to any of the doubles within the tolerance, the statement is true.

If the value of token is not equal to any of the doubles within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple double values are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_eq_dbl WIDTH 3.0 3.5 4.0
user_message "WIDTH is equal to 3.0, 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.2 if\\_tok\\_eq\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.2 if\_tok\_eq\_dbl\_tol

The **if\_tok\_eq\_dbl\_tol** directive tests if a token's **value** is **equal** to any given **double** value, within a given tolerance.

The syntax is:

```
@ if_tok_eq_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if statement is true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token.

At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace.

Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison.

This value must be specified and must be the last value.

If the value of token is equal to any of the doubles within the tolerance, the statement is true.

If the value of token is not equal to any of the doubles within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple double values are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_eq_dbl_tol WIDTH 3.5 0.1
user_message "WIDTH is equal to 3.5 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.3 if\\_tok\\_neq\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.3 if\_tok\_neq\_dbl

The **if\_tok\_neq\_dbl** directive tests if a token's **value** is **not equal** to any of the given **doubles**, within a fixed tolerance of  $1.0 \times 10^{-6}$  (i.e. 1.0E-6).

The syntax is:

```
@ if_tok_neq_dbl <token> <double> [<double>... <double>]
// Commands processed if statement is true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token.

At least one double value must be specified, however, multiple double values can be included.

Multiple double values should be separated by whitespace.

Double values should not be quoted.

If the value of token is not equal to any of the doubles within the tolerance, the statement is true.

If the value of token is equal to any of the doubles within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple double values are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_neq_dbl WIDTH 3.0
user_message "WIDTH is not equal to 3.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.4 if\\_tok\\_neq\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

#### 24.13.11.8.4 if\_tok\_neq\_dbl\_tol

The **if\_tok\_neq\_dbl\_tol** directive tests if a token's **value** is **not equal** to any given **double** values, within a given tolerance.

The syntax is:

```
@ if_tok_neq_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if statement is true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token.

At least one double value must be specified, however, multiple double values can be included.

Multiple double values should be separated by whitespace.

Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison.

This value must be specified and must be the last value.

If the value of token is not equal to any of the double values within the tolerance, the statement is true.

If the value of token is equal to any of the double values within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple double values are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_neq_dbl_tol WIDTH 3.0 0.1
user_message "WIDTH is not equal to 3.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.5 if\\_tok\\_lt\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8.5 if\_tok\_lt\_dbl

The **if\_tok\_lt\_dbl** directive tests if a token is less than any given doubles, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_tok_lt_dbl <token> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

If the token value is less than any of the double values within the tolerance, the statement is true.

If the token value is not less than any of the double values within the tolerance (i.e. the token value is greater than or equal to all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_lt_dbl WIDTH 4.0
user_message "WIDTH is less than 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.6 if\\_tok\\_lt\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.6 if\_tok\_lt\_dbl\_tol

The **if\_tok\_lt\_dbl\_tol** directive tests if a token is less than any given doubles, within a given tolerance. The comparison is numerical.

The syntax is

:

```
@ if_tok_lt_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is less than any of the double values within the tolerance, the statement is true.

If the token value is not less than any of the double values within the tolerance (i.e. the token is greater than or equal to all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_lt_dbl_tol WIDTH 4.0 0.1
user_message "WIDTH is less than 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.7 if\\_tok\\_le\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.7 if\_tok\_le\_dbl

The **if\_tok\_le\_dbl** directive tests if a token is less than or equal to any given doubles, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

:

```
@ if_tok_le_dbl <token> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

If the token value is less than or equal to any of the double values within the tolerance, the statement is true.

If the token value is not less than or equal to any of the double values within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_le_dbl WIDTH 3.5
  user_message "WIDTH is less than or equal to 3.5 within a tolerance of 1e-6"
@ end_if

@ if_tok_le_dbl WIDTH 3.5 4.0
  user_message "WIDTH is less than or equal to 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.8 if\\_tok\\_le\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.8 if\_tok\_le\_dbl\_tol

The `if_tok_le_dbl_tol` directive tests if a token is less than or equal to any given double values, within a given tolerance.

The syntax is:

:

```
@ if_tok_le_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is less than or equal to any of the double values within the tolerance, the statement is true.

If the token value is not less than or equal to any of the double values within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_le_dbl_tol WIDTH 3.5 0.1
  user_message "WIDTH is less than or equal to 3.5 within a tolerance of 0.1"
@ end_if

@ if_tok_le_dbl WIDTH 3.5 4.0 0.1
  user_message "WIDTH is less than or equal to 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.9 if\\_tok\\_gt\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.9 if\_tok\_gt\_dbl

The **if\_tok\_gt\_dbl** directive tests if a token is greater than any given doubles, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

:

```
@ if_tok_gt_dbl <token> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

If the token value is greater than any of the double values within the tolerance, the statement is true.

If the token value is not greater than any of the double values within the tolerance (i.e. the token value is less than or equal to all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_gt_dbl WIDTH 3.0
  user_message "WIDTH is greater than 3.0 within a tolerance of 1e-6"
@ end_if

@ if_tok_gt_dbl WIDTH 3.0 4.0
  user_message "WIDTH is greater than 3.0 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.10 if\\_tok\\_gt\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.10 if\_tok\_gt\_dbl\_tol

The **if\_tok\_gt\_dbl\_tol** directive tests if a token is greater than any given double values, within a given tolerance.

The syntax is:

:

```
@ if_tok_gt_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token.

At least one double value must be specified, however, multiple double values can be included.

Multiple double values should be separated by whitespace.

Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is greater than any of the double values within the tolerance, the statement is true.

If the token value is not greater than any of the double values within the tolerance (*i.e.* the token is less than or equal to all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5
@ if_tok_gt_dbl WIDTH 3.0
  user_message "WIDTH is greater than 3.0 within a tolerance of 1e-6"
@ end_if
@ if_tok_gt_dbl WIDTH 3.0 4.0
  user_message "WIDTH is greater than 3.0 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.11 if\\_tok\\_ge\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8.11 if\_tok\_ge\_dbl

The **if\_tok\_ge\_dbl** directive tests if a token is greater than or equal to any given double values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

:

```
@ if_tok_ge_dbl <token> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

If the token value is greater than or equal to any of the double values within the tolerance, the statement is true.

If the token value is not greater than or equal to any of the double values within the tolerance (i.e. the token value is less than all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_le_dbl WIDTH 3.5
  user_message "WIDTH is less than or equal to 3.5 within a tolerance of 1e-6"
@ end_if

@ if_tok_le_dbl WIDTH 3.5 4.0
  user_message "WIDTH is less than or equal to 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.12 if\\_tok\\_le\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.12 if\_tok\_le\_dbl\_tol

The **if\_tok\_le\_dbl\_tol** directive tests if a token is less than or equal to any given double values, within a given tolerance.

The syntax is:

```
@ if_tok_le_dbl_tol <token> <double> [<double> ... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is less than or equal to any of the double values within the tolerance, the statement is true.

If the token value is not less than or equal to any of the double values within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_le_dbl_tol WIDTH 3.5 0.1
  user_message "WIDTH is less than or equal to 3.5 within a tolerance of 0.1"
@ end_if

@ if_tok_le_dbl WIDTH 3.5 4.0 0.1
  user_message "WIDTH is less than or equal to 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.13 if\\_tok\\_gt\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8.13 if\_tok\_gt\_dbl

The **if\_tok\_gt\_dbl** directive tests if a token is greater than any given doubles, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_tok_gt_dbl <token> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

If the token value is greater than any of the double values within the tolerance, the statement is true.

If the token value is not greater than any of the double values within the tolerance (i.e. the token value is less than or equal to all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_gt_dbl WIDTH 3.0
  user_message "WIDTH is greater than 3.0 within a tolerance of 1e-6"
@ end_if

@ if_tok_gt_dbl WIDTH 3.0 4.0
  user_message "WIDTH is greater than 3.0 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.14 if\\_tok\\_gt\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#) .

#### 24.13.11.8.14 if\_tok\_gt\_dbl\_tol

The **if\_tok\_gt\_dbl\_tol** directive tests if a token is greater than any given double values, within a given tolerance.

The syntax is:

```
@ if_tok_gt_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is greater than any of the double values within the tolerance, the statement is true.

If the token value is not greater than any of the double values within the tolerance (i.e. the token is less than or equal to all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_gt_dbl_tol WIDTH 3.0 0.1
  user_message "WIDTH is greater than 3.5 within a tolerance of 0.1"
@ end_if

@ if_tok_gt_dbl_tol WIDTH 3.0 3.5
  user_message "WIDTH is greater than 3.0 or 3.5 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.15 if\\_tok\\_ge\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8.15 if\_tok\_ge\_dbl

The **if\_tok\_ge\_dbl** directive tests if a token is greater than or equal to any given double values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_tok_ge_dbl <token> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

If the token value is greater than or equal to any of the double values within the tolerance, the statement is true.

If the token value is not greater than or equal to any of the double values within the tolerance (i.e. the token value is less than all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_ge_dbl WIDTH 3.5
user_message "WIDTH is greater than or equal to 3.5 within a tolerance of 1e-6"
@ end_if

@ if_tok_ge_dbl WIDTH 3.5 4.0
user_message "WIDTH is greater than or equal to 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.16 if\\_tok\\_ge\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.16 if\_tok\_ge\_dbl\_tol

The **if\_tok\_ge\_dbl\_tol** directive tests if a token is greater than or equal to any given double values, within a given tolerance.

The syntax is:

```
@ if_tok_ge_dbl_tol <token> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<double>** - the double value to compare against the value of token. At least one double value must be specified, however, multiple double values can be included. Multiple double values should be separated by whitespace. Double values should not be quoted.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is greater than or equal to any of the double values within the tolerance, the statement is true.

If the token value is not greater than or equal to any of the double values within the tolerance (*i.e.* the token value is less than all double values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple doubles are handled as if joined by logical OR statements.

```
@ def_tok WIDTH 3.5

@ if_tok_ge_dbl_tol WIDTH 3.5 0.1
  user_message "WIDTH is greater than or equal to 3.5 within a tolerance of 0.1"
@ end_if

@ if_tok_ge_dbl_tol WIDTH 3.5 4.0
  user_message "WIDTH is greater than or equal to 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.17 if\\_tok\\_in\\_range\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8.17 if\_tok\_in\_range\_dbl

The **if\_tok\_in\_range\_dbl** directive tests if a token is between two double values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_tok_in_range_dbl <token> <lower> <upper>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to test.

**<lower>** - the double representing the lower bound of the range to check against the token. This value must be specified.

**<upper>** - the double representing the upper bound of the range to check against the token. This value must be specified.

If the token value is within the range from lower to upper within the tolerance, the statement is true.

If the token is not defined or is outside the range from lower to upper within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

```
@ def_tok WIDTH 3.5

@ if_tok_in_range_dbl WIDTH 3.0 4.0
  user_message "WIDTH is within range 3.0 to 4.0 within a tolerance of 1e-6"
@ else
  user_message "WIDTH is outside range 3.0 to 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.8.18 if\\_tok\\_in\\_range\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.18 if\_tok\_in\_range\_dbl\_tol

The **if\_tok\_in\_range\_dbl** directive tests if a token is between two double values, within a given tolerance.

The syntax is:

```
@ if_tok_in_range_dbl <token> <lower> <upper> <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<lower>** - the double value representing the lower bound of the range to check against the token. This value must be specified.

**<upper>** - the double value representing the upper bound of the range to check against the token. This value must be specified.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is within the range from lower to upper within the tolerance, the statement is true.

If the token is outside the range from lower to upper within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

```
@ def_tok WIDTH 3.5

@ if_tok_in_range_dbl_tol WIDTH 3.0 4.0 0.1
  user_message "WIDTH is within range 3.0 to 4.0 within a tolerance of 0.1"
@ else
  user_message "WIDTH is outside range 3.0 to 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.8.19 if\\_tok\\_nin\\_range\\_dbl](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.8.19 if\_tok\_nin\_range\_dbl

The **if\_tok\_nin\_range\_dbl** directive tests if a token is not between two double values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_tok_nin_range_dbl <token> <lower> <upper>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<lower>** - the double value representing the lower bound of the range to check against the token. This value must be specified.

**<upper>** - the double value representing the upper bound of the range to check against the token. This value must be specified.

If the token value is within the range from lower to upper within the tolerance, the statement is true.

If the token is outside the range from lower to upper within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

```
@ def_tok WIDTH 4.0

@ if_tok_nin_range WIDTH 3.0 3.5
  user_message "WIDTH is not within range 3.0 to 3.5 within a tolerance of 1e-6"
@ end_if
```

See also:

`if_tok_in_range_dbl` - logically opposite directive

`if_val_nin_range_dbl` - for testing values rather than tokens

Continue to [24.13.11.8.20 if\\_tok\\_nin\\_range\\_dbl\\_tol](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.8.20 if\_tok\_nin\_range\_dbl\_tol

The **if\_tok\_nin\_range\_dbl\_tol** directive tests if a token is between two double values, within a given tolerance.

The syntax is:

```
@ if_tok_nin_range_dbl <token> <lower> <upper> <tolerance>
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<lower>** - the double value representing the lower bound of the range to check against the token. This value must be specified.

**<upper>** - the double value representing the upper bound of the range to check against the token. This value must be specified.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the token value is not within the range from lower to upper within the tolerance, the statement is true.

If the token is within the range from lower to upper within the tolerance, the statement is false.

If the token is undefined, the snippet will produce an error.

```
@ def_tok WIDTH 4.0

@ if_tok_nin_range_dbl_tol WIDTH 3.0 3.5 0.1
  user_message "WIDTH is not within the range 3.0 to 3.5 within a tolerance of 0.1"
@ end_if
```

See also:

if\_tok\_nin\_range\_dbl - for comparison with fixed tolerance

if\_tok\_in\_range\_dbl\_tol - logically opposite directive

if\_val\_nin\_range\_dbl\_tol - for testing of values rather than tokens

Continue to [24.13.11.9 Tokens vs Integers](#) or return to [24.13.11.8 Tokens vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.9 Tokens vs Integers

The following directives compare a token value against given integers.

See

[24.13.11.9.1 if\\_tok\\_eq\\_int](#)

[24.13.11.9.2 if\\_tok\\_neq\\_int](#)

[24.13.11.9.3 if\\_tok\\_lt\\_int](#)

[24.13.11.9.4 if\\_tok\\_le\\_int](#)

[24.13.11.9.5 if\\_tok\\_gt\\_int](#)

[24.13.11.9.6 if\\_tok\\_ge\\_int](#)

Or return to [24.13 Defining and Using Snippets](#).

### 24.13.11.9.1 if\_tok\_eq\_int

The **if\_tok\_eq\_int** directive tests if a token equals any of the given integers.

The syntax is:

```
@ if_tok_eq_int <token> <integer> [<integer>... <integer>]
// Commands to be processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<integer>** - the integer value to compare against the token. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the token value is equal to any of the integers, the statement is true.

If the token value is not equal to any of the integers, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple integers are handled as if joined by logical OR statements.

```
@ def_tok NUM_LANES 2

@ if_tok_eq_int NUM_LANES 2
  user_message "NUM_LANES equals 2"
@ end_if

@ if_tok_eq_int NUM_LANES 2 3 4
  user_message "NUM_LANES equals 2, 3 or 4"
@ end_if
```

See also:

`if_tok_neq_int` - logically opposite directive

`if_tok_eq_dbl` - for comparison with doubles

`if_val_eq_int` - for testing of values rather than tokens

Continue to [24.13.11.9.2 if\\_tok\\_neq\\_int](#) or return to [24.13.11.9 Tokens vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.9.2 if\_tok\_neq\_int

The **if\_tok\_neq\_int** directive tests if a token does not equal any of the integers.

The syntax is:

```
@ if_tok_neq_int <token> <integer> [<integer>... <integer>]
// Commands to be processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<integer>** - the integer value to compare against the token. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the token value does not equal any of the integers, the statement is true.

If the token value equals any of the integers, the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple integers are handled as if joined by logical OR statements.

```
@ def_tok NUM_LANES 2

@ if_tok_neq_int NUM_LANES 2
  user_message "NUM_LANES is not equal to 2"
@ end_if

@ if_tok_neq_int NUM_LANES 2 3 4
  user_message "NUM_LANES equal to 2, 3 or 4"
@ end_if
```

See also:

if\_tok\_eq\_int - logically opposite directive

if\_tok\_neq\_dbl - for comparison with doubles

if\_val\_neq\_int - for testing of values rather than tokens

Continue to [24.13.11.9.3 if\\_tok\\_lt\\_int](#) or return to [24.13.11.9 Tokens vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.9.3 if\_tok\_lt\_int

The **if\_tok\_lt\_int** directive tests if a token is less than any given integers. The comparison is numerical.

The syntax is:

```
@ if_tok_lt_int <token> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<integer>** - the integer value to compare against the token. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the token value is less than any of the integers, the statement is true.

If the token value is not less than any of the integers (*i.e.* the token value is greater than or equal to all integer values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple integers are handled as if joined by logical OR statements.

```
@ def_tok NUM_LANES 2

@ if_tok_lt_int NUM_LANES 3
  user_message "NUM_LANES is less than 3"
@ end_if

@ if_tok_lt_int NUM_LANES 2 3 4
  user_message "NUM_LANES is less than 2, 3 or 4"
@ end_if
```

See also:

if\_tok\_le\_int - for less than or equal to comparison against integers

if\_tok\_gt\_int - for greater than comparison against integers

if\_tok\_ge\_int - for greater than or equal to comparison against integers

if\_tok\_lt\_dbl - for comparison against doubles

if\_val\_lt\_int - for testing of values rather than tokens

Continue to [24.13.11.9.4 if\\_tok\\_le\\_int](#) or return to [24.13.11.9 Tokens vs Integers](#) or [24.13.11 Snippet Directives](#).

#### 24.13.11.9.4 if\_tok\_le\_int

The **if\_tok\_le\_int** directive tests if a token is less than or equal to any given integers. The comparison is numerical.

The syntax is:

```
@ if_tok_le_int <token> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<integer>** - the integer value to compare against the token. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the token value is less than or equal to any of the integers, the statement is true.

If the token value is not less than or equal to any of the integers (i.e. the token value is greater than all integer values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple integers are handled as if joined by logical OR statements.

```
@ def_tok NUM_LANES 2

@ if_tok_le_int NUM_LANES 2
  user_message "NUM_LANES is less than or equal to 2"
@ end_if

@ if_tok_le_int NUM_LANES 2 3 4
  user_message "NUM_LANES is less than or equal to 2, 3 or 4"
@ end_if
```

See also:

[if\\_tok\\_lt\\_int](#) - for less than comparison against integers

[if\\_tok\\_gt\\_int](#) - for greater than comparison against integers

[if\\_tok\\_ge\\_int](#) - for greater than or equal to comparison against integers

[if\\_tok\\_le\\_dbl](#) - for comparison against doubles

[if\\_val\\_le\\_int](#) - for testing of values rather than tokens

Continue to [24.13.11.9.5 if\\_tok\\_gt\\_int](#) or return to [24.13.11.9 Tokens vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.9.5 if\_tok\_gt\_int

The **if\_tok\_gt\_int** directive tests if a token is greater than any given integers. The comparison is numerical.

The syntax is:

```
@ if_tok_gt_int <token> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<integer>** - the integer value to compare against the token. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the token value is greater than any of the integers, the statement is true.

If the token value is not greater than any of the integers (*i.e.* the token value is less than or equal to all integer values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple integers are handled as if joined by logical OR statements.

```
@ def_tok NUM_LANES 3

@ if_tok_neq_int NUM_LANES 2
  user_message "NUM_LANES is greater than 2"
@ end_if

@ if_tok_neq_int NUM_LANES 2 3 4
  user_message "NUM_LANES is greater than 2, 3 or 4"
@ end_if
```

See also:

if\_tok\_le\_int - for less than or equal to comparison against integers

if\_tok\_lt\_int - for less than comparison against integers

if\_tok\_ge\_int - for greater than or equal to comparison against integers

if\_tok\_gt\_dbl - for comparison against doubles

if\_val\_gt\_int - for testing of values rather than tokens

Continue to [24.13.11.9.6 if\\_tok\\_ge\\_int](#) or return to [24.13.11.9 Tokens vs Integers](#) or [24.13.11 Snippet Directives](#) .

## 24.13.11.9.6 if\_tok\_ge\_int

The **if\_tok\_ge\_int** directive tests if a token is greater than or equal to any given integers.

The syntax is:

```
@ if_tok_ge_int <token> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<token>** - the name of the token to compare.

**<integer>** - the integer value to compare against the token. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the token value is greater than or equal to any of the integers, the statement is true.

If the token value is not greater than or equal to any of the integers (*i.e.* the token value is less than all integer values), the statement is false.

If the token is undefined, the snippet will produce an error.

Multiple integers are handled as if joined by logical OR statements.

```
@ def_tok NUM_LANES 2

@ if_tok_ge_int NUM_LANES 2
  user_message "NUM_LANES is greater than or equal to 2"
@ end_if

@ if_tok_ge_int NUM_LANES 2 3 4
  user_message "NUM_LANES is greater than or equal to 2, 3 or 4"
@ end_if
```

See also:

if\_tok\_lt\_int - for less than comparison against integers

if\_tok\_gt\_int - for greater than comparison against integers

if\_tok\_le\_int - for less than or equal to comparison against integers

if\_tok\_ge\_dbl - for comparison against doubles

if\_val\_ge\_int - for testing of values rather than tokens

Continue to [24.13.11.10 Values Defined and Value vs Value](#) or return to [24.13.11.9 Tokens vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.10 Values Defined and Value vs Value

The following directives test whether values are empty and compare values with other values. Values can be parameter values (*i.e.* substituted) or raw values (*i.e.* in the file).

Note: these directives should not be used with tokens. The order in which parameters, directives and tokens are processed and substituted (see section [24.13.13 Order of snippet processing](#)) will likely result in unexpected results.

See

[24.13.11.10.1 if\\_val](#)

[24.13.11.10.2 if\\_nval](#)

[24.13.11.10.3 if\\_all\\_vals](#)

[24.13.11.10.4 if\\_nall\\_vals](#)

Or return to [24.13 Defining and Using Snippets](#).

## 24.13.11.10.1 if\_val

The **if\_val** directive tests if any of the given values are empty.

The syntax is:

```
@ if_val <value> [<value>... <value>]
// Commands processed if statement is true
@ end_if
```

**<value>** - the value to test. At least one value must be specified, however, multiple values can be included. Multiple values should be separated by whitespace.

If any of the values are not empty, the statement is true.

If all of the values are empty, the statement is false.

Multiple values are handled as if joined by logical OR statements.

NOTE: This directive simply tests whether the value is defined and exists and not whether the value is valid or blank.

```
// PARAMETER PA TEXT "PA" OPTIONAL
// PARAMETER PB TEXT "PB" OPTIONAL
// PARAMETER PC TEXT "PC" OPTIONAL

@ if_val "$(PA)"
user_message "Found PA"
@ end_if

@ if_val "$(PA)" "$(PB)"
user_message "Found PA or PB"
@ end_if

@ if_val "$(PA)" "$(PB)" "$(PC)"
user_message "Found PA or PB or PC"

@ end_if
```

Continue to [24.13.11.10.2 if\\_nval](#) or return to [24.13.11.10 Values Defined and Value vs Value](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.10.2 if\_nval

The **if\_nval** directive tests if any of the given values are empty.

The syntax is:

```
@ if_nval <value> [<value>... <value>]
// Commands processed if statement is true
@ end_if
```

**<value>** - the value to test. At least one value must be specified, however, multiple values can be included. Multiple values should be separated by whitespace.

If any of the values are empty, the statement is true.

If none of the values are empty, the statement is false.

Multiple values are handled as if joined by logical OR statements.

```
// PARAMETER PA TEXT "PA" OPTIONAL
// PARAMETER PB TEXT "PB" OPTIONAL
// PARAMETER PC TEXT "PC" OPTIONAL

@ if_nval "$(PA)"
user_message "Not found PA"
@ end_if

@ if_nval "$(PA)" "$(PB)"
user_message "Not found PA and PB"
@ end_if

@ if_nval "$(PA)" "$(PB)" "$(PC)"
user_message "Not found PA,PB and PC"
@ end_if
```

Continue to [24.13.11.10.3 if\\_all\\_vals](#) or return to [24.13.11.10 Values Defined and Value vs Value](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.10.3 if\_all\_vals

The **if\_all\_vals** directive tests whether all given values are not empty.

The syntax is:

```
@ if_all_vals <value> [<value>... <value>]
// Commands to be processed when statement is true
@ end_if
```

**<value>** - the value to test. At least one value must be specified, however, multiple values can be included. Multiple values should be separated by whitespace.

If all of the values are not empty, the statement is true.

If any of the values are empty, the statement is false.

Multiple values are handled as if joined by logical AND statements.

```
// PARAMETER PA TEXT "PA" OPTIONAL
// PARAMETER PB TEXT "PB" OPTIONAL

@ if_all_vals "$(PA)"
  user_message "PA exists"
@ end_if

@ if_all_vals "$(PA)" "$(PB)"
  user_message "PA and PB both exist!"
@ end_if

@ if_all_vals "$(PA)" "$(PB)" "$(PC)"
  user_message "PA, PB and PC all exist!"
@ else
  user_message "One of PA, PB or PC does not exist"
@ end_if
```

Continue to [24.13.11.10.4 if\\_nall\\_vals](#) or return to [24.13.11.10 Values Defined and Value vs Value](#) or [24.13.11 Snippet Directives](#).

#### 24.13.11.10.4 if\_nall\_vals

The **if\_nall\_vals** directive tests if all of the given values are empty.

The syntax is:

```
@ if_nall_vals <value> [<value>... <value>]
// Commands to be processed when statement is true
@ end_if
```

**<value>** - the value to test. At least one value must be specified, however, multiple values can be included. Multiple values should be separated by whitespace.

If all of the values are empty, the statement is true.

If any of the values are not empty, the statement is false.

Multiple values are handled as if joined by logical AND statements.

```
// PARAMETER PA TEXT "PA" OPTIONAL
// PARAMETER PB TEXT "PB" OPTIONAL
// PARAMETER PC TEXT "PC" OPTIONAL

@ if_nall_vals "$(PA)"
user_message "PA does not exist."
@ end_if

@ if_nall_vals "$(PA)" "$(PB)"
user_message "PA and PB do not exist.!"
@ end_if

@ if_nall_vals "$(PA)" "$(PB)" "$(PC)"
user_message "PA,PB and PC do not exist."
@ end_if
```

Continue to [24.13.11.11 Values vs Strings](#) or return to [24.13.11.10 Values Defined and Value vs Value](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.11 Values vs Strings

The following directives compare a value against given strings. Comparison is character-based and not numerical (e.g. "1" does not equal "1.0" or "01").

See

[24.13.11.11.1 if\\_val\\_eq\\_str](#)

[24.13.11.11.2 if\\_val\\_neq\\_str](#)

Or return to [24.13 Defining and Using Snippets](#).

#### 24.13.11.11.1 if\_val\_eq\_str

The **if\_val\_eq\_str** directive tests if a value is equal to any of the given strings.

The syntax is:

```
@ if_val_eq_str <value> <string> [<string>... <string>]
// Commands processed if statement is true
@ end if
```

**<value>** - is the value to compare. One and only one value must be specified.

**<string>** - is the string to compare against the value. At least one string must be specified, however, multiple strings can be included. Multiple strings should be separated by whitespace.

Strings or values containing whitespace should be enclosed in quotation marks.

If the value is equal to any of the strings, the statement is true.

If the value is not equal to any of the string values, the statement is false.

Multiple strings are handled as if joined by logical OR statements.

```
// PARAMETER KT TEXT "Kerb type" "SA"

@ if_val_eq_str "$(KT)" "SA"
    user_message "Kerb type is equal to SA"
@ end_if

@ if_val_eq_str "$(KT)" "SA" "SM"
    user_message "Kerb type is equal to either SA or SM"
@ end_if
```

Continue to [24.13.11.11.2 if\\_val\\_neq\\_str](#) or return to [24.13.11.11 Values vs Strings](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.11.2 if\_val\_neq\_str

The **if\_val\_neq\_str** directive tests if a value is not equal to any of the given strings.

The syntax is:

```
@ if_val_neq_str <value> <string> [<string>... <string>]
// Commands processed if statement is true
@ end if
```

**<value>** - is the value to compare. One and only one value must be specified.

**<string>** - is the string to compare against the value. At least one string must be specified, however, multiple strings can be included. Multiple strings should be separated by whitespace.

Strings or values containing whitespace should be enclosed in quotation marks.

If the value is not equal to any of the strings, the statement is true.

If the value is not equal to none of the strings (*i.e.* it is equal to all strings), the statement is false.

Multiple strings are handled as if joined by logical OR statements.

```
// PARAMETER KT TEXT "Kerb type" "SB"

@ if_val_neq_str "${KT}" "SA"
  user_message "Kerb type is not equal to SA"
@ end_if

@ if_val_neq_str "${KT}" "SA" "SM"
  user_message "Kerb type is not equal to either SA or SM"
@ end_if
```

Continue to [24.13.11.12 Values vs Doubles](#) or return to [24.13.11.11 Values vs Strings](#) or [24.13.11 Snippet Directives](#) .

## 24.13.11.12 Values vs Doubles

The following directives compare a token value against given doubles. Comparison is numerical.

See

- [24.13.11.12.1 if\\_val\\_eq\\_dbl](#)
- [24.13.11.12.2 if\\_val\\_eq\\_dbl\\_tol](#)
- [24.13.11.12.3 if\\_val\\_neq\\_dbl](#)
- [24.13.11.12.4 if\\_val\\_neq\\_dbl\\_tol](#)
- [24.13.11.12.5 if\\_val\\_lt\\_dbl](#)
- [24.13.11.12.6 if\\_val\\_lt\\_dbl\\_tol](#)
- [24.13.11.12.7 if\\_val\\_le\\_dbl](#)
- [24.13.11.12.8 if\\_val\\_le\\_dbl\\_tol](#)
- [24.13.11.12.9 if\\_val\\_gt\\_dbl](#)
- [24.13.11.12.10 if\\_val\\_gt\\_dbl\\_tol](#)
- [24.13.11.12.11 if\\_val\\_ge\\_dbl](#)
- [24.13.11.12.12 if\\_val\\_ge\\_dbl\\_tol](#)
- [24.13.11.12.13 if\\_val\\_in\\_range\\_dbl](#)
- [24.13.11.12.14 if\\_val\\_in\\_range\\_dbl\\_tol](#)
- [24.13.11.12.15 if\\_val\\_nin\\_range\\_dbl](#)
- [24.13.11.12.16 if\\_val\\_nin\\_range\\_dbl\\_tol](#)

Or return to [24.13 Defining and Using Snippets](#).

### 24.13.11.12.1 if\_val\_eq\_dbl

The **if\_val\_eq\_dbl** directive tests if a value is equal to any of the given double values, with a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_eq_dbl <value> <double> [<double>... <double>]
// Commands processed if statement is true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

Values containing whitespace should be enclosed in quotation marks.

If the value is equal to any of the doubles within the tolerance, the statement is true.

If the value is not equal to any of the doubles within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_eq_dbl $(WIDTH) 3.5
  user_message "WIDTH is equal to 3.5 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.2 if\\_val\\_eq\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.12.2 if\_val\_eq\_dbl\_tol

The **if\_val\_eq\_dbl\_tol** directive tests if a value is equal to any given double value, within a given tolerance.

The syntax is:

```
@ if_val_eq_dbl_tol <value> <double> [<double>... <double>] <tolerance>
// Commands processed if statement is true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is equal to any of the double values within the tolerance, the statement is true.

If the value is not equal to any of the double values within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple double values are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_eq_dbl_tol $(WIDTH) 3.5 0.1
  user_message "WIDTH is equal to 3.5 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.3 if\\_val\\_neq\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.3 if\_val\_neq\_dbl

The **if\_val\_neq\_dbl** directive tests if a value is not equal to any of the given doubles, with a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_neq_dbl <value> <double> [<double>... <double>]
// Commands processed if statement is true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

If the value is not equal to any of the doubles within the tolerance, the statement is true.

If the value is equal to all of the doubles within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_neq_dbl $(WIDTH) 3.0
  user_message "WIDTH is not equal to 3.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.4 if\\_val\\_neq\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.12.4 if\_val\_neq\_dbl\_tol

The **if\_val\_neq\_dbl\_tol** directive tests if a value is not equal to any given doubles, within a given tolerance.

The syntax is:

```
@ if_val_neq_dbl_tol <value> <double> [<double>... <double>] <tolerance>
// Commands processed if statement is true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is not equal to any of the doubles within the tolerance, the statement is true.

If the value is equal to all of the doubles within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple double values are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_neq_dbl_tol $(WIDTH) 3.0 0.1
  user_message "WIDTH is not equal to 3.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.5 if\\_val\\_lt\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.5 if\_val\_lt\_dbl

The **if\_val\_lt\_dbl** directive tests if a value is less than any of the given double values within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_lt_dbl <value> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

If the value is less than any of the doubles within the tolerance, the statement is true.

If the value is not less than any of the doubles within the tolerance (*i.e.* the value is greater than or equal to all doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_lt_dbl $(WIDTH) 4.0
  user_message "WIDTH is less than 4.0 within a tolerance of 1e-6"
@ end_if

@ if_val_lt_dbl $(WIDTH) 3.0 3.5 4.0
  user_message "WIDTH is less than 3.0, 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.6 if\\_val\\_lt\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#) .

## 24.13.11.12.6 if\_val\_lt\_dbl\_tol

The **if\_val\_lt\_dbl\_tol** directive tests if a value is less than any given double values, within a given tolerance.

The syntax is:

```
@ if_val_lt_dbl_tol <value> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is less than any of the doubles within the tolerance, the statement is true.

If the value is not less than any of the doubles within the tolerance (*i.e.* the value is greater than or equal to all doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_lt_dbl_tol $(WIDTH) 4.0 0.1
  user_message "WIDTH is less than 4.0 within a tolerance of 0.1"
@ end_if

@ if_val_lt_dbl_tol $(WIDTH) 3.0 3.5 4.0 0.1
  user_message "WIDTH is less than 3.0, 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.7 if\\_val\\_le\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.7 if\_val\_le\_dbl

The **if\_val\_le\_dbl** directive tests if a value is less than or equal to any of the given double values within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_le_dbl <value> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

If the value is less than or equal to any of the doubles within the tolerance, the statement is true.

If the value is not less than or equal to any of the doubles within the tolerance (*i.e.* the value is greater than all doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_le_dbl $(WIDTH) 3.5
  user_message "WIDTH is less than or equal to 3.5 within a tolerance of 1e-6"
@ end_if

@ if_val_le_dbl $(WIDTH) 3.0 3.5 4.0
  user_message "WIDTH is less than or equal to 3.0, 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.8 if\\_val\\_le\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#) .

## 24.13.11.12.8 if\_val\_le\_dbl\_tol

The **if\_val\_le\_dbl\_tol** directive tests if a value is less than or equal to any given doubles, within a given tolerance.

The syntax is:

```
@ if_val_le_dbl_tol <value> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is less than or equal to any of the doubles within the tolerance, the statement is true.

If the value is not less than or equal to any of the doubles within the tolerance (*i.e.* the value is greater than all of the doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_le_dbl_tol $(WIDTH) 3.5 0.1
  user_message "WIDTH is less than or equal to 3.5 within a tolerance of 0.1"
@ end_if

@ if_val_le_dbl_tol $(WIDTH) 3.0 3.5 4.0 0.1
  user_message "WIDTH is less than or equal to 3.0, 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.9 if\\_val\\_gt\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.9 if\_val\_gt\_dbl

The **if\_val\_gt\_dbl** directive tests if a value is greater than any of the given doubles, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_gt_dbl <value> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

If the value is greater than any of the doubles within the tolerance, the statement is true.

If the value is not greater than any of the doubles within the tolerance (*i.e.* the value is less than or equal to all doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_gt_dbl $(WIDTH) 3.0
  user_message "WIDTH is greater than 3.5 within a tolerance of 1e-6"
@ end_if

@ if_val_gt_dbl $(WIDTH) 3.0 3.5 4.0
  user_message "WIDTH is greater than 3.0, 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.8 if\\_val\\_le\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.12.10 if\_val\_gt\_dbl\_tol

The **if\_val\_gt\_dbl\_tol** directive tests if a value is greater than any given double values, within a given tolerance.

The syntax is:

```
@ if_val_gt_dbl_tol <value> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is greater than any of the doubles within the tolerance, the statement is true.

If the value is not greater than any of the doubles within the tolerance (*i.e.* the value is less than or equal to all doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_gt_dbl_tol $(WIDTH) 3.0 0.1
  user_message "WIDTH is greater than 3.0 within a tolerance of 0.1"
@ end_if

@ if_val_gt_dbl_tol $(WIDTH) 3.0 3.5 4.0 0.1
  user_message "WIDTH is greater than 3.0, 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.11 if\\_val\\_ge\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.11 if\_val\_ge\_dbl

The **if\_val\_ge\_dbl** directive tests if a value is greater than or equal to any of the given double values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_ge_dbl <value> <double> [<double>... <double>]
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

If the value is greater than or equal to any of the doubles within the tolerance, the statement is true.

If the value is not greater than or equal to any of the doubles within the tolerance(*i.e.* the value is less than all of the double values), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_ge_dbl $(WIDTH) 3.5
  user_message "WIDTH is greater than or equal to 3.5 within a tolerance of 1e-6"
@ end_if

@ if_val_gt_dbl $(WIDTH) 3.0 3.5 4.0
  user_message "WIDTH is greater than or equal to 3.0, 3.5 or 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.12 if\\_val\\_ge\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#) .

## 24.13.11.12.12 if\_val\_ge\_dbl\_tol

The **if\_val\_ge\_dbl\_tol** directive tests if a value is greater than or equal to any given double values, within a given tolerance.

The syntax is:

```
@ if_val_ge_dbl_tol <value> <double> [<double>... <double>] <tolerance>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value - integer or double.

**<double>** - the double to compare against the value. At least one double must be specified, however, multiple double can be included. Multiple double should be separated by whitespace.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is greater than or equal to any of the double within the tolerance, the statement is true.

If the value is not greater than or equal to any of the doubles within the tolerance (*i.e.* the value is less than all doubles), the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

Multiple doubles are handled as if joined by logical OR statements.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_ge_dbl_tol $(WIDTH) 3.0 0.1
  user_message "WIDTH is greater than or equal to 3.0 within a tolerance of 0.1"
@ end_if

@ if_val_ge_dbl_tol $(WIDTH) 3.0 3.5 4.0 0.1
  user_message "WIDTH is greater than or equal to 3.0, 3.5 or 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.13 if\\_val\\_in\\_range\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.13 if\_val\_in\_range\_dbl

The **if\_val\_in\_range\_dbl** directive tests if a value exists and is between two double values, within a fixed tolerance of  $1.0 \times 10^{-6}$  (1.0E-6).

The syntax is:

```
@ if_val_in_range_dbl <value> <lower> <upper>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<lower>** - the double representing the lower bound of the range to check against value. This value must be specified.

**<upper>** - the double representing the upper bound of the range to check against value. This value must be specified.

If the value is within the range from lower to upper within the tolerance, the statement is true.

If the value is not within the range from lower to upper within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_in_range_dbl $(WIDTH) 3.0 4.0
  user_message "WIDTH is between 3.0 and 4.0 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.14 if\\_val\\_in\\_range\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.12.14 if\_val\_in\_range\_dbl\_tol

The **if\_val\_in\_range\_dbl\_tol** directive tests if a value exists and is between two doubles, within a given tolerance.

The syntax is:

```
@ if_val_in_range_dbl_tol <value> <lower> <upper> <tolerance>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<lower>** - the double value representing the lower bound of the range to check against value. This value must be specified.

**<upper>** - the double value representing the upper bound of the range to check against value. This value must be specified.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is within the range from lower to upper within the tolerance, the statement is true.

If the value is not within the range from lower to upper within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

```
// PARAMETER WIDTH REAL "Lane width" 3.5

@ if_val_in_range_dbl_tol $(WIDTH) 3.0 4.0 0.1
  user_message "WIDTH is between 3.0 and 4.0 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.12.15 if\\_val\\_nin\\_range\\_dbl](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.12.15 if\_val\_nin\_range\_dbl

The **if\_val\_nin\_range\_dbl** directive tests if a value exists and is not between two double values, within a given tolerance.

The syntax is:

```
@ if_val_nin_range_dbl <value> <lower> <upper>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<lower>** - the double representing the lower bound of the range to check against value. This value must be specified.

**<upper>** - the double representing the upper bound of the range to check against value. This value must be specified.

If the value is not within the range from lower to upper within the tolerance, the statement is true.

If the value is within the range from lower to upper within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

```
// PARAMETER WIDTH REAL "Lane width" 4.0

@ if_val_nin_range_dbl $(WIDTH) 3.0 3.5
  user_message "WIDTH is outside the range 3.0 to 3.5 within a tolerance of 1e-6"
@ end_if
```

Continue to [24.13.11.12.16 if\\_val\\_nin\\_range\\_dbl\\_tol](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.12.16 if\_val\_nin\_range\_dbl\_tol

The **if\_val\_nin\_range\_dbl\_tol** directive tests if a value exists and is not between two double values, within a given tolerance.

The syntax is:

```
@ if_val_nin_range_dbl_tol <value> <lower> <upper> <tolerance>
// Commands processed if true
@ end_if
```

**<value>** - the value to compare. The value should evaluate to a numeric value- integer or double.

**<lower>** - the double representing the lower bound of the range to check against value. This value must be specified.

**<upper>** - the double representing the upper bound of the range to check against value. This value must be specified.

**<tolerance>** - the tolerance, expressed as a valid double value, for comparison. This value must be specified and must be the last value.

If the value is not within the range from lower to upper within the tolerance, the statement is true.

If the value is within the range from lower to upper within the tolerance, the statement is false.

If the value or any doubles do not evaluate to a valid number, an error is produced.

```
// PARAMETER WIDTH REAL "Lane width" 4.0

@ if_val_nin_range_dbl_tol $(WIDTH) 3.0 3.5 0.1
  user_message "WIDTH is outside the range 3.0 to 3.5 within a tolerance of 0.1"
@ end_if
```

Continue to [24.13.11.13 Values vs Integers](#) or return to [24.13.11.12 Values vs Doubles](#) or [24.13.11 Snippet Directives](#) .

### 24.13.11.13 Values vs Integers

The following directives compare a value against given integers.

See

- [24.13.11.13.1 if\\_val\\_eq\\_int](#)
- [24.13.11.13.2 if\\_val\\_neq\\_int](#)
- [24.13.11.13.3 if\\_val\\_eq\\_all\\_ints](#)
- [24.13.11.13.4 if\\_val\\_neq\\_all\\_ints](#)
- [24.13.11.13.5 if\\_val\\_lt\\_int](#)
- [24.13.11.13.6 if\\_val\\_le\\_int](#)
- [24.13.11.13.7 if\\_val\\_gt\\_int](#)
- [24.13.11.13.8 if\\_val\\_ge\\_int](#)
- [24.13.11.13.9 if\\_val\\_lt\\_all\\_ints](#)
- [24.13.11.13.10 if\\_val\\_le\\_all\\_ints](#)
- [24.13.11.13.11 if\\_val\\_gt\\_all\\_ints](#)
- [24.13.11.13.12 if\\_val\\_ge\\_all\\_ints](#)

Or return to [24.13 Defining and Using Snippets](#).

#### 24.13.11.13.1 if\_val\_eq\_int

The **if\_val\_eq\_int** directive tests if a value equals any of the given integers.

The syntax is:

```
@ if_val_eq_int <value> <integer> [<integer>... <integer>]
// Commands to be processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is equal to any of the integers, the statement is true.

If the value is not equal to any of the integers, the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical OR statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_eq_int $(NUM_LANES) 2
  user_message "NUM_LANES is equal to 2"
@ end_if
```

Continue to [24.13.11.13.2 if\\_val\\_neq\\_int](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.13.2 if\_val\_neq\_int

The **if\_val\_neq\_int** directive tests if a value is not equal to any of the integers.

The syntax is:

```
@ if_val_neq_int <value> <integer> [<integer>... <integer>]
// Commands to be processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is not equal to any of the integers, the statement is true.

If the value is equal to all of the integers, the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical OR statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_neq_int $(NUM_LANES) 3
  user_message "NUM_LANES is not equal to 3"
@ end_if
```

Continue to [24.13.11.13.3 if\\_val\\_eq\\_all\\_ints](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.13.3 if\_val\_eq\_all\_ints

The **if\_val\_eq\_all\_ints** directive tests if a value is equal to all of the given integers.

The syntax is:

```
@ if_val_eq_all_ints <value> <integer> [<integer>... <integer>]
// Commands to be processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is equal to all of the integers, the statement is true.

If the value is not equal to all of the integers, the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical AND statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_eq_all_ints $(NUM_LANES) 2
  user_message "NUM_LANES is equal to 2"
@ end_if

@ if_val_eq_all_ints $(NUM_LANES) 2 3 4
  user_message "NUM_LANES is equal to 2, 3 and 4" // This should never be true
@ end_if
```

Continue to [24.13.11.13.4 if\\_val\\_neq\\_all\\_ints](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#) .

## 24.13.11.13.4 if\_val\_neq\_all\_ints

The **if\_val\_neq\_all\_ints** directive tests if a value is not equal to all of the given integers.

The syntax is:

```
@ if_val_neq_all_ints <value> <integer> [<integer>... <integer>]
// Commands to be processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is not equal to all of the integer values, the statement is true.

If the value is equal to any of the integer values, the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical AND statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_neq_all_ints $(NUM_LANES) 3 4 5
  user_message "NUM_LANES is not equal to 3, 4 AND 5"
@ end_if
```

Continue to [24.13.11.13.5 if\\_val\\_lt\\_int](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.13.5 if\_val\_lt\_int

The **if\_val\_lt\_int** directive tests if a value is less than any given integers.

The syntax is:

```
@ if_val_lt_int <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is less than any of the integers, the statement is true.

If the value is not less than any of the integers (*i.e.* the value is greater than or equal to all integer values), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical OR statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_lt_int $(NUM_LANES) 3 4
  user_message "NUM_LANES is less than 3 or 4"
@ end_if
```

See also:

if\_val\_le\_int - for less than or equal to comparison against integers

if\_val\_gt\_int - for greater than comparison against integers

if\_val\_ge\_int - for greater than or equal to comparison against integers

if\_val\_lt\_dbl - for comparison against doubles

if\_tok\_lt\_int - for testing of tokens rather than values

Continue to [24.13.11.13.6 if\\_val\\_le\\_int](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.13.6 if\_val\_le\_int

The **if\_val\_le\_int** directive tests if a value is less than or equal to any given integers.

The syntax is:

```
@ if_val_le_int <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is less than or equal to any of the integers, the statement is true.

If the value is not less than or equal to any of the integers (*i.e.* the value is greater than all integer values), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical OR statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_le_int $(NUM_LANES) 3 4
  user_message "NUM_LANES is less than or equal to 3 or 4"
@ end_if
```

See also:

if\_val\_lt\_int - for less than comparison against integers

if\_val\_gt\_int - for greater than comparison against integers

if\_val\_ge\_int - for greater than or equal to comparison against integers

if\_val\_le\_dbl - for comparison against doubles

if\_tok\_le\_int - for testing of tokens rather than values

Continue to [24.13.11.13.7 if\\_val\\_gt\\_int](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.13.7 if\_val\_gt\_int

The **if\_val\_gt\_int** directive tests if a value is greater than any given integers.

The syntax is:

```
@ if_val_gt_int <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is greater than any of the integers, the statement is true.

If the value is not greater than any of the integers (*i.e.* the value is less than or equal to all integer values), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical OR statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 5

@ if_val_gt_int $(NUM_LANES) 3 4
  user_message "NUM_LANES is greater than 3 or 4"
@ end_if
```

See also:

if\_val\_le\_int - for less than or equal to comparison against integers

if\_val\_lt\_int - for less than comparison against integers

if\_val\_ge\_int - for greater than or equal to comparison against integers

if\_val\_gt\_dbl - for comparison against doubles

if\_tok\_gt\_int - for testing of tokens rather than values

Continue to [24.13.11.13.8 if\\_val\\_ge\\_int](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.13.8 if\_val\_ge\_int

The **if\_val\_ge\_int** directive tests if a value is greater than or equal to any given integers.

The syntax is:

```
@ if_val_ge_int <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is greater than or equal to any of the integers, the statement is true.

If the value is not greater than or equal to any of the integers (*i.e.* the value is less than all integer values), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical OR statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 3

@ if_val_ge_int $(NUM_LANES) 3 4
  user_message "NUM_LANES is greater than or equal to 3 or 4"
@ end_if
```

See also:

if\_val\_lt\_int - for less than comparison against integers

if\_val\_gt\_int - for greater than comparison against integers

if\_val\_le\_int - for less than or equal to comparison against integers

if\_val\_ge\_dbl - for comparison against doubles

if\_tok\_ge\_int - for testing of tokens rather than values

Continue to [24.13.11.13.9 if\\_val\\_lt\\_all\\_ints](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.13.9 if\_val\_lt\_all\_ints

The **if\_val\_lt\_all\_ints** directive tests if a value is less than all given integers. The comparison is numerical.

The syntax is:

```
@ if_val_lt_all_ints <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is less than all of the integers, the statement is true.

If the value is not less than all of the integers (*i.e.* the value is greater than or equal to any integer values), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical AND statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 2

@ if_val_lt_all_ints $(NUM_LANES) 3 4 5
  user_message "NUM_LANES is less than 3, 4 AND 5"
@ end_if
```

See also:

if\_val\_le\_all\_ints- for less than or equal to comparison against integers

if\_val\_gt\_all\_ints - for greater than comparison against integers

if\_val\_ge\_all\_ints - for greater than or equal to comparison against integers

if\_val\_lt\_int - for comparison against any integers

Continue to [24.13.11.13.10 if\\_val\\_le\\_all\\_ints](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.13.10 if\_val\_le\_all\_ints

The **if\_val\_le\_all\_ints** directive tests if a value is less than or equal to all given integers.

The syntax is:

```
@ if_val_le_all_ints <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is less than or equal to all of the integers, the statement is true.

If the value is not less than or equal to all of the integers (*i.e.* the value is greater than any integer), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical AND statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 3

@ if_val_le_all_ints $(NUM_LANES) 3 4 5
  user_message "NUM_LANES is less than or equal to 3, 4 AND 5"
@ end_if
```

See also:

if\_val\_lt\_all\_ints - for less than comparison against integers

if\_val\_gt\_all\_ints - for greater than comparison against integers

if\_val\_ge\_all\_ints - for greater than or equal to comparison against integers

if\_val\_le\_int - for comparison against any integers

Continue to [24.13.11.13.11 if\\_val\\_gt\\_all\\_ints](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

### 24.13.11.13.11 if\_val\_gt\_all\_ints

The **if\_val\_gt\_all\_ints** directive tests if a value is greater than all given integers. The syntax is:  
The syntax is:

```
@ if_val_gt_all_ints <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is greater than all of the integers, the statement is true.

If the value is not greater than all of the integers (*i.e.* the value is less than or equal to any integer), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical AND statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 5

@ if_val_gt_all_ints $(NUM_LANES) 2 3 4
  user_message "NUM_LANES is greater than 2, 3 AND 4"
@ end_if
```

See also:

if\_val\_le\_all\_ints - for less than or equal to comparison against integers

if\_val\_lt\_all\_ints - for less than comparison against integers

if\_val\_ge\_all\_ints - for greater than or equal to comparison against integers

if\_val\_gt\_int - for comparison against any integers

Continue to [24.13.11.13.12 if\\_val\\_ge\\_all\\_ints](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.11.13.12 if\_val\_ge\_all\_ints

The **if\_val\_ge\_all\_ints** directive tests if a value is greater than or equal to all given integers.

The syntax is:

```
@ if_val_ge_all_ints <value> <integer> [<integer>... <integer>]
// Commands processed if true
@ end_if
```

**<value>** - the value to test. The value should evaluate to a valid integer.

**<integer>** - the integer value to compare against the value. At least one integer must be specified, however, multiple integers can be included. Multiple integers should be separated by whitespace.

If the value is greater than or equal to all of the integers, the statement is true.

If the value is not greater than or equal to all of the integers (*i.e.* the value is less than any integer values), the statement is false.

If the value or any integers do not evaluate to a valid integer, an error is produced.

Multiple integers are handled as if joined by logical AND statements.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 5

@ if_val_ge_all_ints $(NUM_LANES) 3 4 5
  user_message "NUM_LANES is greater than or equal to 3, 4 AND 5"
@ end_if
```

See also:

if\_val\_lt\_all\_ints - for less than comparison against integers

if\_val\_gt\_all\_ints - for greater than comparison against integers

if\_val\_le\_all\_ints - for less than or equal to comparison against integers

if\_val\_ge\_int - for comparison against any integers

Continue to [24.13.12 Snippet Variables](#) or return to [24.13.11.13 Values vs Integers](#) or [24.13.11 Snippet Directives](#).

## 24.13.12 Snippet Variables

In **12d Model 11** snippets, it is possible to define variables whose values are calculated from the design model. For example, we can capture the offset or width of a link as a variable. This allows snippets to be more flexible and work with normal modifier commands that can alter the position of links.

See

- [24.13.12.1 Defining Link Variables](#)
- [24.13.12.2 Defining General Variables](#)
- [24.13.12.3 Using Helper Functions](#)
- [24.13.12.4 Using Variables in Other Variables](#)
- [24.13.12.5 Using Variables in Commands](#)

Or return to [24.13 Defining and Using Snippets](#).

### 24.13.12.1 Defining Link Variables

The first necessary step in using snippet link variables is to define them. It is assumed that we already have some existing links in our design- not necessarily in the snippet or even output to a model. Defining a link variable allow 12d to identify which particular links need to be measured and calculated.

The syntax to define a link variable is:

```
link_variable <link_name> <chg_range> <alias>
```

**<link\_name>** is the shorthand in MTF code for picking the desired link from the model. Refer to Sect. 19.1.1.2 of the 12d Help for more info.

**<chg\_range>** is the chainage range for this variable, typically the standard range for the entire snippet: `$_SCH) 0 $_ECH) 0`

**<alias>** is the name for this particular link\_variable. We will be able to refer to this link\_variable elsewhere in our snippet by this name.

A link\_variable is tied to a particular link in the MTF model. It is simply an alias to a particular link in our design.

So, for example, if we have a link called KLIP on the Design layer that we want to refer to, we would do this:

```
link_variable "Design=>KLIP" $_SCH) 0 $_ECH) 0 "LINK1"
```

"Design=>KLIP" is the link "KLIP" from the "Design" layer. The side (left/right) will be automatically selected depending on which side this snippet is applied.

`$_SCH) 0 $_ECH) 0` is the standard snippet text defining the chainage range

"LINK1" is our custom alias for this particular link.

Continue to [24.13.12.2 Defining General Variables](#) or return to [24.13.12 Snippet Variables](#)

### 24.13.12.2 Defining General Variables

In addition to link variables, which are specifically tied to a particular link in the model, it is possible to define general variables for intermediate calculations or storage of values. These are done with the generic variable command.

The general variable command has the syntax:

```
variable <name> <chg_range> <value>
```

**<name>** is the name for this variable and must be in quotes, e.g. "VAR1"

**<chg\_range>** is the chainage range for this variable, typically the standard range for the entire snippet: \$\_SCH) 0 \$\_ECH) 0

**<value>** is the value for this variable, which can include other variables, directives and formulae.

For example:

```
// PARAMETER WIDTH REAL "Road width" 3.5
variable "HALF_W" $_SCH) 0 $_ECH) 0 "($_WIDTH) / 2"
```

So, if the WIDTH parameter had a value of 3.5, the resulting variable HALF\_W would have a calculated value of  $3.5 / 2 = 1.75$ . This value is calculated at run-time and at every section to which this snippet applies.

Continue to [24.13.12.3 Using Helper Functions](#) or return to [24.13.12 Snippet Variables](#)

### 24.13.12.3 Using Helper Functions

Snippets often require the calculation of various values and measurements. For example, the current offset of a link from the centreline or the height difference between two links.

These measurements and values could be used for performing conditional tests or simply as the basis for other parts of the snippet.

To this end, there are various helper functions available within snippets. 12d will automatically calculate the relevant value at every applicable section every time it is run. These can, therefore, also be used within variable commands.

```
link_variable "Design=>KLIP" $_SCH) 0 $_ECH) 0 "LINK1"  
variable "HALF_W" $_SCH) 0 $_ECH) 0 "(link_width(LINK1) / 2)"
```

In this example, the value of the variable "HALF\_W" is calculated from the width of the link\_variable with the alias "LINK1" divided by 2. This way, if another modifier or snippet modified the link, we could still use it within our snippet without the user needing to manually enter any width manually.

The list of available helper function is:

link\_width(XX) - the width of the link XX relative to the previous link

link\_height(XX) - the height of the link XX relative to the previous link

link\_grade(XX) - the grade of the link XX in y/x (not in %) relative to the previous link

rel\_offset(XX) - the unsigned offset of the link XX, typically +ve for LHS and RHS

abs\_offset(XX) - the signed offset of the link XX, typically -ve for LHS and +ve for RHS

abs\_height(XX) - the signed height difference from the hinge of the link XX, -ve lower, +ve higher

abs\_level(XX) - the level of the link XX, equal to (hinge level + height difference)

x\_coord(XX) - the X ordinate of the link XX

y\_coord(XX) - the Y ordinate of the link XX

apply\_tin\_height(XX) - the height of the Apply MTF tin at the given link XX

other\_tin\_height(LINK;TIN) - to capture the height of any tin (TIN) at a link (LINK) (note the semicolon separating the captured link and the tin name).

named\_grade(NG) - Capture signed grade (m/m) of a named grade, NG

named\_grade\_hdsd\_fact(NG) - For named grade, NG, the factor to multiply a horizontal distance by to get a slope distance.

named\_grade\_sdhd\_fact(NG) - For named grade, NG, the factor to multiply a slope distance by to get a horizontal distance.

named\_grade\_zdsd\_fact(NG) - For named grade, NG, the factor to multiply a height difference by to get a slope distance.

named\_grade\_sdzd\_fact(NG) - For named grade, NG, the factor to multiply a slope distance by to get a height difference.

named\_grade\_hdzd\_fact(NG) - For named grade, NG, the factor to multiply a horizontal distance

by to get a height difference.

named\_grade\_zdhd\_fact(NG) - For named grade, NG, the factor to multiply a height difference by to get a horizontal distance.

More helper functions may be added in the future.

Continue to [24.13.12.4 Using Variables in Other Variables](#) or return to [24.13.12 Snippet Variables](#)

#### 24.13.12.4 Using Variables in Other Variables

Once variables have been defined, they can be used and substituted into other variables.

To do so, wherever you would like to insert the variable, use the syntax:

```
variable(name)
```

Where variable is the keyword and name is the name of the variable to be substituted.

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 1
// PARAMETER LANE_WIDTH REAL "Lane width" 3.5
// PARAMETER XFALL "Crossfall %" -3

variable "W0" $_SCH) 0 $_ECH) 0 "($(LANE_WIDTH) * $(NUM_LANES))"
variable "H0" $_SCH) 0 $_ECH) 0 "(variable(W0) * $(XFALL) / 100)"
```

Continue to [24.13.12.5 Using Variables in Commands](#) or return to [24.13.12 Snippet Variables](#)

### 24.13.12.5 Using Variables in Commands

Once variables have been defined, they can be used in a selection of modifier commands. To do so, wherever you would like to insert the variable, use the syntax:

```
variable <name>
```

Where variable is a keyword and name is the name of the variable to be substituted.

For example:

```
// PARAMETER NUM_LANES INTEGER "Number of lanes" 1
// PARAMETER LANE_WIDTH REAL "Lane width" 3.5

variable "W0" $_SCH) 0 $_ECH) 0 "$ (LANE_WIDTH) * $(NUM_LANES))"

insert "Design=>EDGE" "red" variable W0 unknown -3.0 $_SCH) 0 $_ECH) 0 absolute extra_start
extra_end
```

Note that an MTF modifier commands must be specifically written to support the use of variables. Currently, the MTF modifier commands and their values that support variables are:

Modify\_Decision\_Batter: width, height, slope

Modify\_Decision\_Batter\_Test: width, height, slope

Modify\_XFall\_Point: m\_offset, m\_height1

Modify\_Insert\_Absolute: width, height, slope

Modify\_Insert\_Link: width, height, slope

Modify Link All

Modify\_All\_To\_2\_RL

More may be added in the future.

Continue to [24.13.13 Order of snippet processing](#) or return to [24.13.12 Snippet Variables](#)

## 24.13.13 Order of snippet processing

When a snippet is used in **12d Model**, it is as part of an MTF.

Each Snippet referred to in the MTF is in its own file, and each snippet file goes through set steps to calculate and substitute values in the snippet, before it is used in the MTF file.

It is important to understand each step in the processing of a snippet file, and the order of the steps, to ensure that snippet runs as you expect it to.

See

[Step 1. Initial Loading of the Snippet File](#)

[Step 2. Parameters from the MTF Snippet Panel Are Substituted](#)

[Step 3. Snippet Directives Executed](#)

[Step 4. Token Values Substituted](#)

[Step 5. Snippet Inserted into MTF](#)

[Step 6. C Preprocessor Run as Part of MTF processing](#)

Or return to [24.13 Defining and Using Snippets](#).

## Step 1. Initial Loading of the Snippet File

The snippet file is read and loaded into memory.

All comments and blank lines are removed.

An example of a snippet file that we will use at each step is:

```
// PARAMETER FRUIT TEXT "Enter a fruit"
```

```
@ def_tok FRUIT_TOKEN "$(FRUIT)"
```

```
@ if_val_eq_str "$(FRUIT)" "apple"
```

```
  user_message "Correct, you entered apple!"
```

```
@ else
```

```
  user_message "You entered $(FRUIT), I wanted an apple!"
```

```
@ end_if
```

```
@ if_val_eq_str "$(FRUIT_TOKEN)" "apple"
```

```
  user_message "Correct, you entered apple!"
```

```
@ else
```

```
  user_message "You entered $(FRUIT_TOKEN), I wanted an apple!"
```

```
@ end_if
```

**Warning** - this may not do what you first think it will as we will see shortly.



Continue to [Step 2. Parameters from the MTF Snippet Panel Are Substituted](#) or return to [24.13.13 Order of snippet processing](#).

## Step 2. Parameters from the MTF Snippet Panel Are Substituted

In the next step on the snippet file, any values of parameters that are specified in the **MTF Snippet** panel. are substituted.

That is, all instances of parameter variables, e.g. \$TOK or \$(TOK), are replaced with the value that was selected by the user in the **MTF Snippet** panel.

Automatic parameters (e.g. \$\_SCH)) are also substituted.

If a parameter cannot be substituted, for example, it has not been defined, it is left as-is in the temporary **MTF** file and an error to the user will be produced.

As an example of the parameter substitution, if in the snippet given in **Step 1**, the user entered a value of **apple** (or accepted the default value) for the **Enter a fruit** parameter field on the **MTF Snippet** panel, then after the Parameter Substitution for FRUIT, the snippet would then look like:

```
@ def_tok FRUIT_TOKEN "apple"

@ if_val_eq_str "apple" "apple"
  user_message "Correct, you entered apple!"
@ else
  user_message "You entered apple, I wanted an apple!"
@ end_if

@ if_val_eq_str "$(FRUIT_TOKEN)" "apple"
  user_message "Correct, you entered apple!"
@ else
  user_message "You entered $(FRUIT_TOKEN), I wanted an apple!"
@ end_if
```

Continue to [Step 3. Snippet Directives Executed](#) or return to [24.13.13 Order of snippet processing](#)

### Step 3. Snippet Directives Executed

In the next pass of the snippet file, any snippet directives are parsed and executed.

At this point, no parameters (or automatic parameters) exist, because they have all been substituted in the previous processing step.

So the snippet started in **Step 1**, will now look like:

```
user_message "Correct, you entered apple!"
user_message "You entered $(FRUIT_TOKEN), I wanted an apple!"
```

Notice that the flow-control previously present has been evaluated and only the applicable commands (in this case, `user_message` commands) remain.

#### How did we get the second line?

The second conditional test directive:

```
@ if_val_eq_str "$(FRUIT_TOKEN)" "apple"
```

has evaluated to **false**, because in `if_val_eq_str` the value `"$(FRUIT_TOKEN)"` is being treated as the raw text string `$(FRUIT_TOKEN)` which is not equal to **apple**.

Hence the corresponding **@ else** block of commands is used and we get the line

```
user_message "You entered $(FRUIT_TOKEN), I wanted an apple!"
```

is left in the snippet. Which is probably not what you wanted.

In this particular case, the snippet author would have been better off using the **@ if\_tok\_eq\_str** directive instead of the **@ if\_val\_eq\_str** directive.

```
@ if_tok_eq_str FRUIT_TOKEN "apple"
user_message "Correct, you entered apple!"
@ else
user_message "You entered $(FRUIT_TOKEN), I wanted an apple!"
@ end_if
```

The conditional test directive:

```
@ if_tok_eq_str FRUIT_TOKEN "apple"
```

will be evaluated to **true**, because `FRUIT_TOKEN` is **apple** and we then get the line in the snippet:

```
user_message "Correct, you entered apple!"
```

Continue to [Step 4. Token Values Substituted](#) or return to [24.13.13 Order of snippet processing](#).

#### Step 4. Token Values Substituted

In the next processing stage of the snippet, tokens have their values substituted, in a similar manner to that in which parameters were substituted in the earlier step.

At this point, this is simply a replacement of tokens in the snippet file, \$TOKEN or \$(TOKEN), with their value.

There is no flow control at this point, since it has already been performed in the previous step.

So the snippet started in **Step 1**, will now look like:

```
user_message "Correct, you entered apple!"  
user_message "You entered apple, I wanted an apple!"
```

Note that each line containing tokens is processed as many times as necessary until no further tokens remain to be substituted. This allows token values to contain other tokens.

```
@ def_tok TOK1 1.0  
@ def_tok TOK2 2.0  
@ def_tok TOK3 "($(TOK1) + $(TOK2))"// TOK3  
will eventually equal 3.0
```

Although the substitution of parameters and tokens is similar, it is important to understand they occur at separate stages of the snippet processing.

After Step 4, the **temporary** MTF snippet file is generated (.temp\_mtf).

All intelligence, formulae and user input have been removed in the temporary MTF snippet file.

Continue to [Step 5. Snippet Inserted into MTF](#) or return to [24.13.13 Order of snippet processing](#)

## Step 5. Snippet Inserted into MTF

At this stage, whatever remains of the snippet is inserted into the MTF File. No further processing is done by the 12d Snippet Preprocessor.

However, note, that the older C preprocessor- the one that handles #define, #ifdef, etc.- has not yet been run.

Continue to [Step 6. C Preprocessor Run as Part of MTF processing](#) or return to [24.13.13 Order of snippet processing](#).

## Step 6. C Preprocessor Run as Part of MTF processing

In the next pass, the C preprocessor- i.e. the one handling #define, #if, etc.- is used on the entire MTF, which now includes the interpreted and processed snippets.

Any parts of any snippets still using the older C preprocessor directives (#define, #if, etc) will be processed at this stage as part of the overall MTF. At this point there is no concept of a snippet as a separate entity- all snippet results have already been inserted into the MTF.

### Note

Because of the power and predictability of the **Snippet Directives**, they should always be used instead of any C Preprocessor commands.

For more information on the substitutions and C preprocessor directives supported in the MTF, refer to the **12d Model Reference** manual.

Return to [24.13.13 Order of snippet processing](#) or [24.13 Defining and Using Snippets](#).

## 24.13.14 Debugging Snippets

See

[24.13.14.1 Print Messages and Log Lines to the Output Window](#)

[24.13.14.2 Intermediate Parsing Results](#)

[24.13.14.3 Temporary MTF Snippet File](#)

Or return to [24.13 Defining and Using Snippets](#).

### 24.13.14.1 Print Messages and Log Lines to the Output Window

To help with writing and debugging snippets, V11 provides the ability to write messages to the Output Window. These are done via special snippet commands *user\_message\_print*, *user\_message\_log* and *user\_message* with the syntax:

(a) *user\_message\_print*

**user\_message\_print** text\_msg "variable" "var\_name" st\_ch extra\_st end\_ch extra\_end

"variable" - optional.

"var\_name" is the name of a variable.

"variable" is optional but if it exists then "var\_name" must also exist.

**st\_ch extra\_st end\_ch extra\_end** - optional

**st\_ch** is a chainage

**extra\_st** is added to st\_ch to give the **start chainage**

**end\_ch** is a chainage

**extra\_end** is added to end\_ch to give the **end chainage**

Print the text **text\_msg** to the Output Window and then print the value of the variable *var\_name* to the Output Window.

The messages are produced for every section in the chainage range given by the start chainage and the end chainage.

If the start and end chainage modes are omitted, they are taken to be from the start to the end of the reference string.

```
user_message_print "Value of REL_OFFSET: " "variable" "REL_OFFSET" $_(SCH) 0 $_(ECH) 0
```

For every section between the start and the end chainage, this will print the text "Value of REL\_OFFSET " followed by the value of the variable REL\_OFFSET to the Output Window,

(b) *user\_message\_log*, *user\_message*

**user\_message\_log** text\_msg "variable" "var\_name" st\_ch extra\_st end\_ch extra\_end

**user\_message** text\_msg "variable" "var\_name" st\_ch extra\_st end\_ch extra\_end

"variable" - optional.

"var\_name" is the name of a variable.

"variable" is optional but if it exists then "var\_name" must also exist.

**st\_ch extra\_st end\_ch extra\_end** - optional

**st\_ch** is a chainage

**extra\_st** is added to st\_ch to give the **start chainage**

**end\_ch** is a chainage

**extra\_end** is added to end\_ch to give the **end chainage**

Create a log line in the Output Window of the text **text\_msg** followed by the value of the variable *var\_name*.

The log lines are produced for every section in the chainage range given by the start chainage and the end chainage.

If the start and end chainage modes are omitted, they are taken to be from the start to the end of the reference string.

```
user_message_log "Value of REL_OFFSET: " "variable" "REL_OFFSET" $_SCH) 0 $_ECH) 0
```

For every section between the start and the end chainage, a log line will be created in the Output Window with the text "Value of REL\_OFFSET " followed by the value of the variable REL\_OFFSET

**TIP:** If you only want one message, rather than a message for every section, use:

```
$_SCH) 0 ($_SCH) 0.0001
```

```
user_message_print "This will only print once!" $_SCH) 0 $_SCH) 0.0001
```

Continue to the next section [24.13.14.2 Intermediate Parsing Results](#) or return to [24.13.14 Debugging Snippets](#) or [24.13 Defining and Using Snippets](#).

### 24.13.14.2 Intermediate Parsing Results

As explained in the section Order of Snippet Processing, any MTF Snippet file is processed in several rounds, with each round of the process performing various parsing, evaluation and substitution steps. To assist with debugging and understanding the manner in which snippets are processed by **12d Model**, there is an additional snippet instruction that can be used.

```
// OUTPUT_PARSING_STAGES <filename>
```

For example:

```
// OUTPUT_PARSING_STAGES "dump.txt"
```

With this instruction present in a snippet, 12d will write a detailed log of the snippet at each stage of processing to the given file. The contents of the file are updated/overwritten on each recalc of the Apply MTF function.

For compiled snippets, even if this instruction is present, no output will be produced.

Continue to the next section [24.13.14.3 Temporary MTF Snippet File](#) or return to [24.13.14 Debugging Snippets](#) or [24.13 Defining and Using Snippets](#).

### 24.13.14.3 Temporary MTF Snippet File

To assist with the development and debugging of MTF snippets, there are a number of environment variables to control the naming of, and stop the deletion of, the temporary MTF file that is created after the Snippet Directives are evaluated and the snippet parameters have been substituted (see [Snippet Directives](#)).

At the point of the temporary MTF snippet file being created (.mtfsnippet.temp\_mtf), all intelligence has been removed from the output. All formulae, variables, directives and fixed decisions have been removed and only the raw modifier commands remain.

#### (a) Don't Delete Snippet Temp File

MTF\_SNIPPET\_TEMP\_FILE\_USE\_SNIPPET\_NAME\_KEEP\_4D 1/0

If enabled (non zero), the temporary MTF file that is generated when MTF snippets are processed and parsed will not be deleted. Each run of the **Apply MTF** function will create or update the contents of a file containing the snippet after processing. The file will be created in the current project's working directory.

#### (b) Snippet Temp File Use Snippet Name

MTF\_SNIPPET\_TEMP\_FILE\_USE\_SNIPPET\_NAME\_4D 1/0

If enabled (non zero), the temporary snippet file generated when MTF snippets are processed will be given the name of the snippet file with the extension ".temp\_mtf".

For example, if the snippet file is called "INSERT\_KERB.mtfsnippet" then, by default and when enabled, the temporary snippet file produced will be "INSERT\_KERB.mtfsnippet.temp\_mtf".

This env variable enables the user to have this consistently named temporary file open in an editor and it can then be refreshed each time a snippet is run.

#### (c) Snippet Temp File Custom Extension

MTF\_SNIPPET\_TEMP\_FILE\_EXTENSION\_4D *Extension\_text*

If ***Extension\_text*** is **not blank**, the temporary snippet file generated when MTF snippets have been processed will have the extension *Extension\_text* rather than the default extension of "temp\_mtf".

#### (d) Unique Snippet Temp File Name

MTF\_SNIPPET\_TEMP\_FILE\_UNIQUE\_4D 1/0

If enabled (non zero), a temporary snippet file will be generated for each MTF Snippet instance in the corresponding MTF. To better relate the temporary snippet file with the actual instance in the MTF, this setting will produce a temporary snippet file for every inserted snippet and be named like so:

<Snippet\_File> - <MTF> <side> <modifier\_number>.<extension>

For example,

CP.mtfsnippet - RSCR01.mtf Left No. 7.temp\_mtf

Corresponds to the CP.mtfsnippet file inserted in the left side modifiers of RSCR01.mtf file as modifier number 7.

These environment variables are defined in the **MTF & Boxing >MTF General** section of the **env.4d Editor** which is brought up by selecting **Project =>Management =>env.4d**. See [\\_MTF](#)

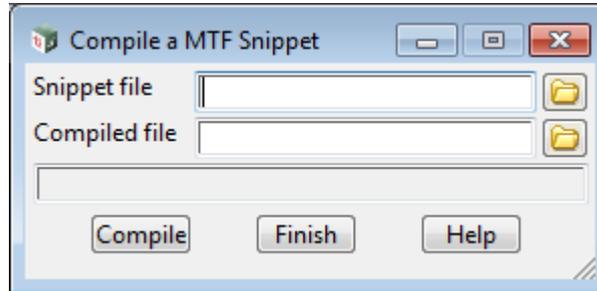
[& Boxing > MTF General](#) in [8 Project](#) .

Continue to the next section [24.13.15 Compiling Snippets](#) or return to [24.13.14 Debugging Snippets](#) or [24.13 Defining and Using Snippets](#) .

## 24.13.15 Compiling Snippets

It is possible to compile a snippet via the option

**Design =>MTF =>Snippet =>Compile snippet**



This can be done to protect any intellectual property (IP) and minimise concerns people may have in developing and sharing snippets.

Compiling a snippet encrypts and encodes the snippet contents into a new file with an extension of **mtfsnippetc** (note the 'c' at the end, denoting a compiled snippet).

A compiled snippet can be run and used by anyone, but the snippet code itself cannot be viewed or edited.

Temporary snippet files, if enabled, are not generated for compiled snippets.

### Original Snippet

```
// INFO Example compiled snippet
```

### Compiled Snippet

```
nyKgSBIhX6wGqQizY/fRfDcT/YReG2J34wsghywz6Ys+cfVn+Pi+crz/4c9XgNbYI+kmS14F
flh+l/bNu41RYyD4RtNvliS9
```

Continue to the next section [24.13.16 Tips and Tricks](#) or return to [24.13.14 Debugging Snippets](#) or [24.13 Defining and Using Snippets](#).

## 24.13.16 Tips and Tricks

- When creating snippets, create the MTF modifier commands first and then copy and paste into a snippet. This helps ensure the format and syntax of the modifier commands is correct.
- Use comments liberally to document your snippet. This makes it easier for you and anyone else to see and understand your thought processes, reasoning and logic within the snippet, making it easier to debug and modify.
- Define tokens for commonly-used values to make code more readable and portable.

```
@ def_tok ABS "absolute"
@ def_tok ES "extra_start"
@ def_tok EE "extra_end"
@ def_tok ASE "$ (ABS) $ (ES) $ (EE)"
Insert "LINK" "red" 3.5 unknown -3 $ (_SCH) 0 $ (_ECH) 0 $ (ASE)
```

- When writing or testing snippets, output messages using `user_message` commands with values of parameters, variables and tokens to check correct calculation of values. These can be commented out or removed before finalising your snippet.
- The `user_message` commands will, by default, output a message for every applicable Apply MTF section. To cut down on the number of repetitive messages, specify a start and end chainage.

```
// PARAMETER WIDTH REAL "Link width" 1.0
@ def_tok MSG_CH "$ (_SCH) 0 $ (_SCH) 0.0001"// Define a token for easier readability
user_message "Link width = $ (WIDTH)" $ (MSG_CH)// Only output one message
```

- For all directives, there is a space between the `@` and the name of the directive!
- For expressions and formula, use parentheses as much as needed. If in doubt, add more and double-check the results
- Parameter names should not contain spaces and should not start with an underscore.
- Token names should not contain spaces and should not start with an underscore.
- It is recommended that token names only contain alphanumeric characters and underscores.
- Parameter names and token names must be unique within the snippet file!
- Use the new V11 syntax, where possible! The V10 syntax- e.g. `#define`, `$VAL` - is deprecated and only supported for backwards-compatibility.

Continue to the next section [24.13.17 Major Warning - You Will be Caught by This](#) or return to [24.13 Defining and Using Snippets](#).

## 24.13.17 Major Warning - You Will be Caught by This

**WARNING** - once a snippet with parameters has been saved in a MTF file, the parameters for the snippet are saved **in the MTF file**. Then when the Apply MTF is run, the parameters and their values are read in from the MTF file and passed into the current snippet file.

Hence if the snippet file has been modified and parameter definitions added, modified or removed since the snippet was originally placed in the MTF, these modifications of parameters will be **ignored** until the snippet panel is reopened in the MTF editor which forces the snippet file to be parsed again and the changes to the parameter definitions are then recognised.

Note that when the Apply MTF for the MTF is run, the MTF used the **current** snippet file so if the snippet file has been modified since the snippet was placed in the MTF, the **modified** snippet file will be used (apart from any new parameter definitions and defaults)

Return to [24.13 Defining and Using Snippets](#).

# 25. Plot Menu

See

[25.1 Zero Padding in Plot File Names](#)

[25.2 New Plot to Models Node](#)

[25.3 Extra Underscore in Long Sect Plot Names](#)

[25.4 Title Blocks and External Data Sources](#)

[25.5 Plot Sheet](#)

## 25.1 Zero Padding in Plot File Names

For the Plot Parameter Files for

- Cross sections
- Long sections
- Plot frames
- Drainage
- Melbourne Water
- Pipeline

There is a new optional parameter called ***Digits in plot file number***, on the main page of each PPF editor.

Leaving ***Digits in plot file number*** blank or setting a value of 1 or less, will result in file names like we've always known them.

Setting ***Digits in plot file number*** to a value of 2 or more, will ensure at least that number of digits in the page numbers built into the plot file names, zero-padding where necessary.

The image shows a screenshot of a software dialog box titled "Plotter parameters". It contains three input fields with corresponding icons on the right: "Plotter type" with a printer icon, "Plot file stem" with a folder icon, and "Digits in plot file number" with a numeric keypad icon. The "Digits in plot file number" field is circled in red. The "Plotter type" field contains the text "model".

## 25.2 New Plot to Models Node

There is a new **Plot to models** node.

The **Plot to models** node controls whether models are cleaned before the plot is written to them, and whether the plots go to separate models for each page or all into one model.

Note that the panel field **Clean plot models beforehand** has been moved from the front panel to the **Plot to models** node.

### *Section: Options for plotting to models*

Panel field	Parameter name	Type	
<b>Clean plot models beforehand</b>	plot_model_clean	choice box	do not clean
			prompt for clean
			always clean

*when plotting to a model, it defines whether to clean (delete the elements in) any resultant plot models that may already exist, before generating the plot(s).*

*This parameter is only applicable if plotting to a model or models. Note that if the models are cleaned using this parameter, any non-plot or locked elements found in the models will not be cleaned from the models, and the plot job will be cancelled.*

### *Section: Translate and merge models*

*Whenever a PPF is set to produce multiple sheets of plot models, the **Translate and merge plot models** group of parameters, offer the ability to produce plot models arranged neatly in a row or column of sheets, optionally merged into the single plot model created for the first sheet.*

*Note: The **Origin X** and **Origin Y** parameters are useful when building up larger arrays of plot sheets - - perhaps entire drawing sets -- from multiple PPFs, where each PPF may be specified with a different origin.*

<b>Mode</b>	plot_model_translate_mode	choice box	do not translate
			row wise translation
			column wise translation

*The entire group of parameters is activated by setting the Mode to either "row-wise translation" or "column-wise translation". This determines whether the multiple sheets will appear as a row or a column of sheets. If unspecified, "do not translate or merge" is assumed, and the remaining parameters are ignored*

<b>Spacing (mm)</b>	plot_model_translate_spacing	real box
---------------------	------------------------------	----------

*determines how far to translate each sheet from the previous sheet. Practically, it should be at least as large as the sheet width or height. For an A1 sheet size, 1000 mm works well as a nice round number*

<b>Origin X (mm)</b>	plot_model_translate_origin_x	real box
----------------------	-------------------------------	----------

*determines the x-coordinate of the bottom-left corner of the first sheet.*

*If let blank then 0 mm is assumed.*

<b>Origin Y (mm)</b>	plot_model_translate_origin_y	real box
----------------------	-------------------------------	----------

*determines the y-coordinate of the bottom-left corner of the first sheet.*

*If let blank then 0 mm is assumed.*

<b>Merge</b>	plot_model_merge	tick box
--------------	------------------	----------

*if **ticked**, all sheets produced by the PPF, will be created in a single model (and that model is the plot model created for the first sheet).*

## 25.3 Extra Underscore in Long Sect Plot Names

When plotting a model of primary strings via the Long-section PPF, an underscore is now automatically inserted between the primary string name and the page number in the plot file name. This underscore will be converted to a space if plotting to a model.

## 25.4 Title Blocks and External Data Sources

See

[25.4.1 Ad Hoc Queries](#)

[25.4.2 Drawing Register in Plot PPF's](#)

### 25.4.1 Ad Hoc Queries

It is possible to extract information from some external data sources to use in User Title Block variables, or directly in the title block file, to place in the title blocks of plots.

Data can be extracted from

- (a) Excel spreadsheets
- (b) XML files
- (c) Databases supporting ODBC

The external data can be accessed in a completely user definable way, known as an **Ad Hoc Queries**.

For each different data source, there is a different method to specify how to identify the bit of information needed from the database.

See

[25.4.1.1 EXCEL](#)

[25.4.1.2 XML](#)

[25.4.1.3 ODBC \(Open Database Connectivity\)](#)

#### 25.4.1.1 EXCEL

The Excel spreadsheet to extract information from **must** have the **first row as a header row** with the names of each column in it so row (n+1) is only the n'th row of actual information.

The format to retrieve the cell X(n+1) from the Sheet **sheet\_n** is

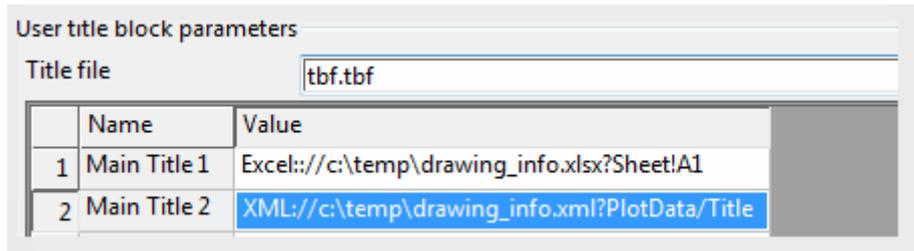
Excel://Full\_Path\_To\_Excel\_File?sheet\_n?Xn

It is Xn is because the first row of the spreadsheet is the header row so Xn is referring to the n'th row of actual information but that is the (n+1) row in the spreadsheet.

For example,

Excel://c:\temp\drawing.info.xlsx?Sheet1?A1

will open the Excel file **c:\temp\drawing.info.xlsx** and get Cell **A2** from the sheet **Sheet 1**.



### 25.4.1.2 XML

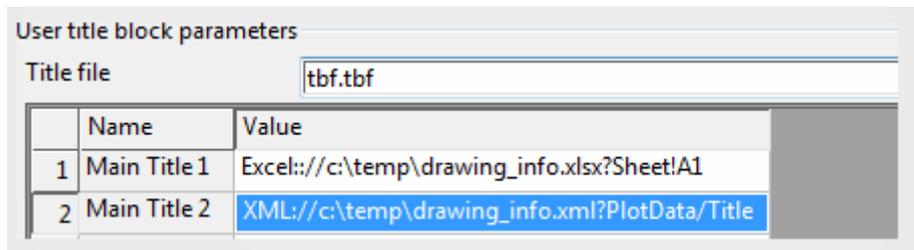
The format to retrieve the value of the element **sub\_elt** inside the element **elt** is

XML://Full\_Path\_To\_XML\_File?elt/sub\_elt

For example,

XML:///c:/temp/drawing\_info.xml?PlotData/Title

will open the XML file **c:\temp\drawing\_info.xml** and extract the value of the Title element below the element PlotData.



So from the xml file

```
<PlotData>
  <Title>My Title</Title>
  <Client>12d</Client>
  etc
</PlotData>
```

it will get the value "My Title"

### 25.4.1.3 ODBC (Open Database Connectivity)

The format to retrieve information from a database is

ODBC://ConnectionString?Query

Unfortunately the format of the **ConnectionString** and the **Query** to pull data from a database is totally dependent on the database system.

But it would resemble something like

ODBC://DSN=MyDBDsn?SELECT Title FROM PlotData WHERE PlotType = 'long'

**ODBC** is for advanced users only, and the user must be familiar with the requirements of the end database system.

## 25.4.2 Drawing Register in Plot PPF's

Many people use database tables, such as an Excel spreadsheet, to provide information for the title blocks of the plots. For example, special drawing numbers and drawing revision. The database is often known as a *Drawing Register*.

For example, in an Excel spreadsheet, the columns of the spreadsheet would have unique names to refer to the information in each column and each row of the spreadsheet holds the information for a single, specific plot.

	A	B	C	D
1	PLOT FILE	DRG#	TITLE	REV
2	plot_xs_MC01_1	12345678-5-1	CROSS SECTIONS MC01	C
3	plot_xs_MC01_2	12345678-5-2	CROSS SECTIONS MC01	C
4	plot_xs_MC01_3	12345678-5-3	CROSS SECTIONS MC01	C
5	plot_xs_MC02_1	12345678-5-4	CROSS SECTIONS MC02	C
6	plot_xs_MC02_2	12345678-5-5	CROSS SECTIONS MC02	C
7	plot_ls_MC01_1	12345678-6-1	LONGITUDINAL SECTION MC01	C
8	plot_ls_MC01_2	12345678-6-2	LONGITUDINAL SECTION MC01	C
9	plot_ls_MC02_1	12345678-6-3	LONGITUDINAL SECTION MC02	C
10	plot_dr_ls_1	12345678-7-1	DRAINAGE LONGITUDINAL SECTIONS	C
11	plot_dr_ls_2	12345678-7-2	DRAINAGE LONGITUDINAL SECTIONS	C
12	plot_dr_ls_3	12345678-7-3	DRAINAGE LONGITUDINAL SECTIONS	C
13				
14				
15				

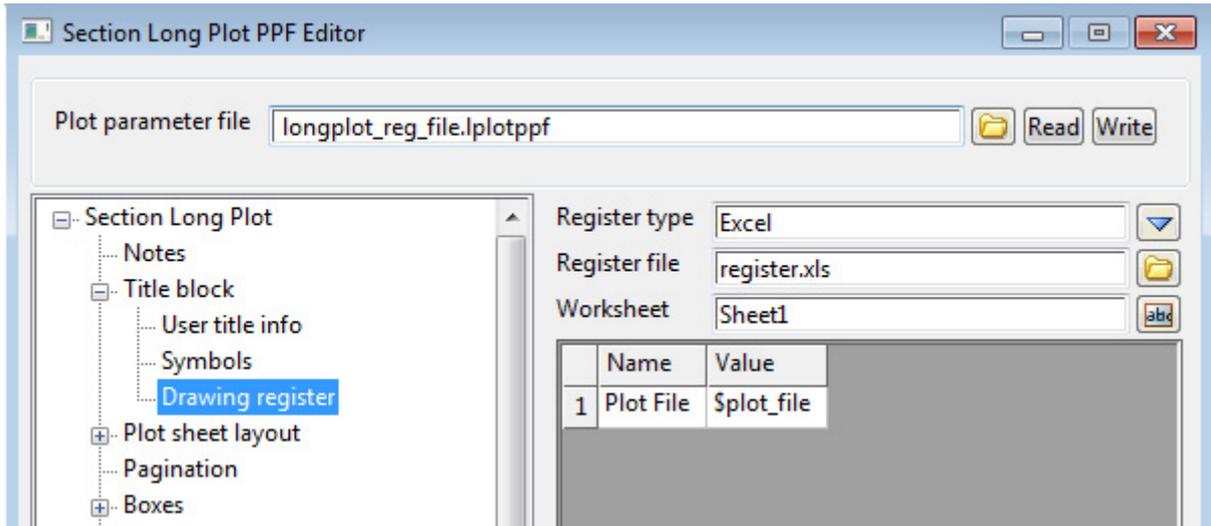
In **12d Model** plot parameter files, there is a **Drawing Register** node which specifies the type and name of the database to be used to retrieve information from, and enough information to determine what row of the database table is to be used to supply information for the title block of a specific plot.

For example in the spreadsheet above, the *plot file name* will identify a particular plot, and the information in each of the columns from the rest of the row is the drawing number (DRG#), Title (TITLE) and Revision (REV) for that plot.

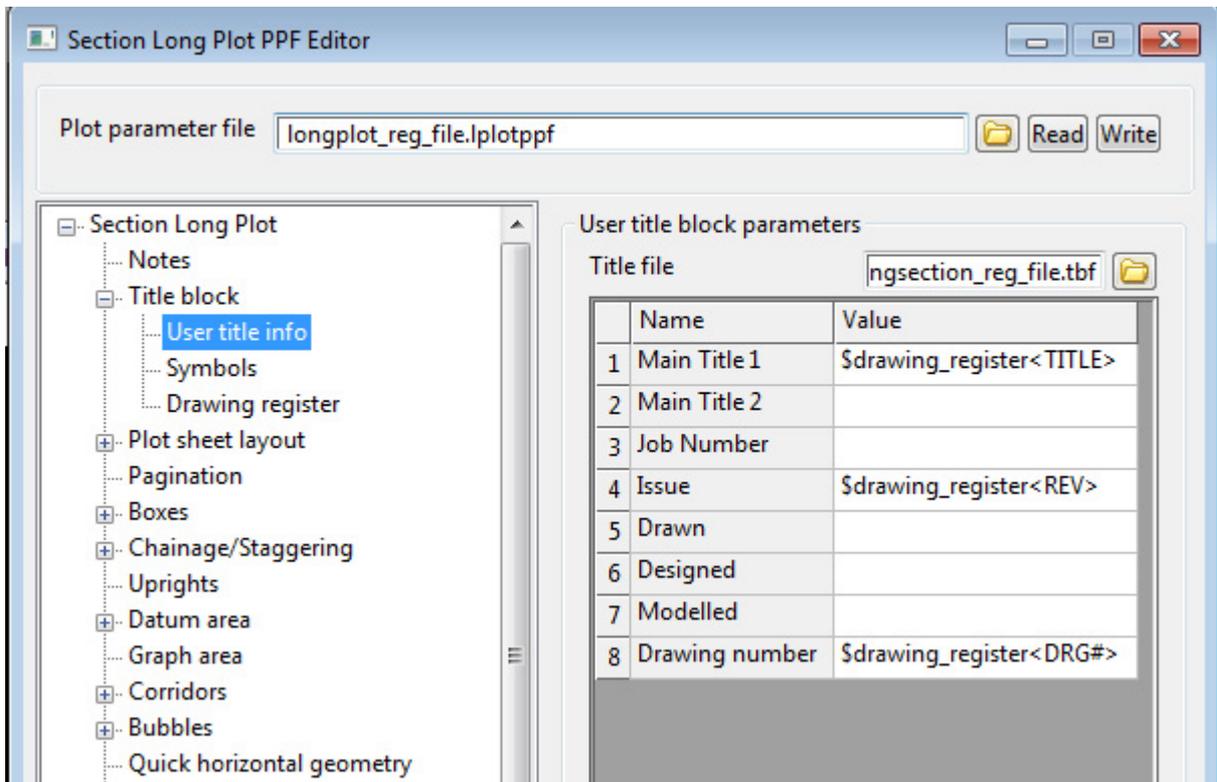
Once the **Drawing Register** node has been set up to identify a row of the spreadsheet, the title block variable to extract the information for column **column\_name** from that row of the drawing register is

**\$drawing\_register<column\_name>**

So the Drawing register node of a Long Section Plot PPF could be



and this could be used in a **User tile info** node as:



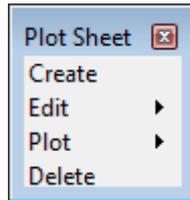
## 25.5 Plot Sheet

Position of menu: **Plot =>Plot sheet**

The **Plot sheet** option is for defining the layout of plotting areas on a sheet of paper and to produce a pdf file of what is on the sheet of paper.

That is **Plot Sheet** is for setting up plot compositions. For V11 it is only for setting up areas from plan views.

The **Plot Sheet** menu is:



See

[25.5.1 Plot Sheet Create](#)

[25.5.2 Plot Sheet Edit](#)

[25.5.3 Plot Sheet Plot](#)

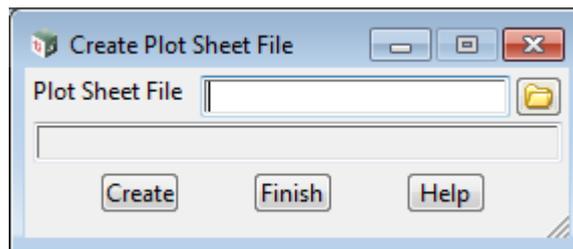
[25.5.4 Plot Sheet Delete](#)

### 25.5.1 Plot Sheet Create

**Current position on menu:** **Plot =>Plot sheet =>Create**

**Plot sheet create** is for creating a new plot sheet file.

Selecting **Create** brings up the **Create Plot Sheet File** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Plot sheet file</b>	file box		*.12dpsf files

*name for the file to create.*

*This file name must not already exist in the working folder.*

**Create** button

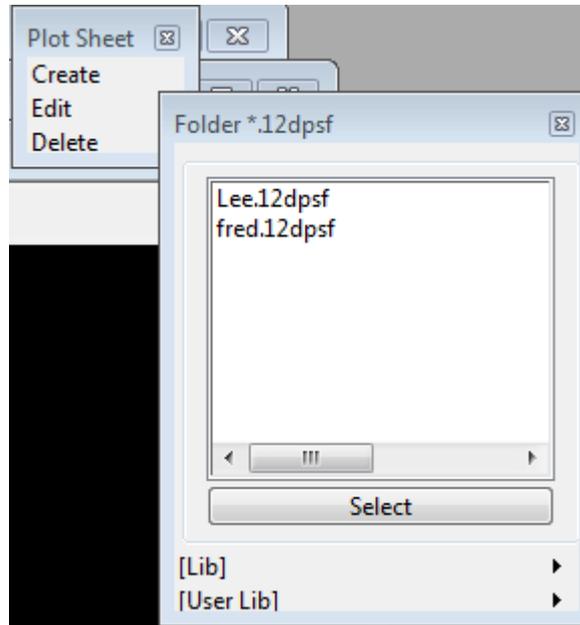
*when pressed, a new plot sheet file is created with the name given in the **Plot sheet file** field and then the **Plot Sheet Create/Editor** panel is started with that file as the Plot Sheet File being edited. See [25.5.2.1 Plot Sheet Create/Edit](#).*

## 25.5.2 Plot Sheet Edit

**Current position on menu:** Plot =>Plot sheet =>Edit

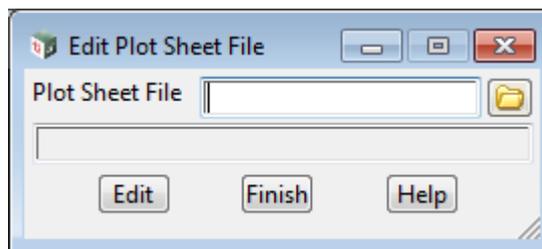
The **Edit** option works in two ways - you can click on **Edit** itself or walk right on **Edit** and see a list of all the plot sheet files in the working folder.

The **Edit** walk right lists all the plot sheet files in the working folder.



Selecting a plot sheet file from the list, or from another area, brings up the **Plot Sheet Create/Editor** panel already loaded with the selected plot sheet file.

By clicking on **Edit** itself, the **Edit Plot Sheet File** panel is brought up.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Plot sheet file</b>	file box		*.12dpsf files

*name for the plot sheet file to edit.  
This file name must exist in the working folder.*

**Edit** button

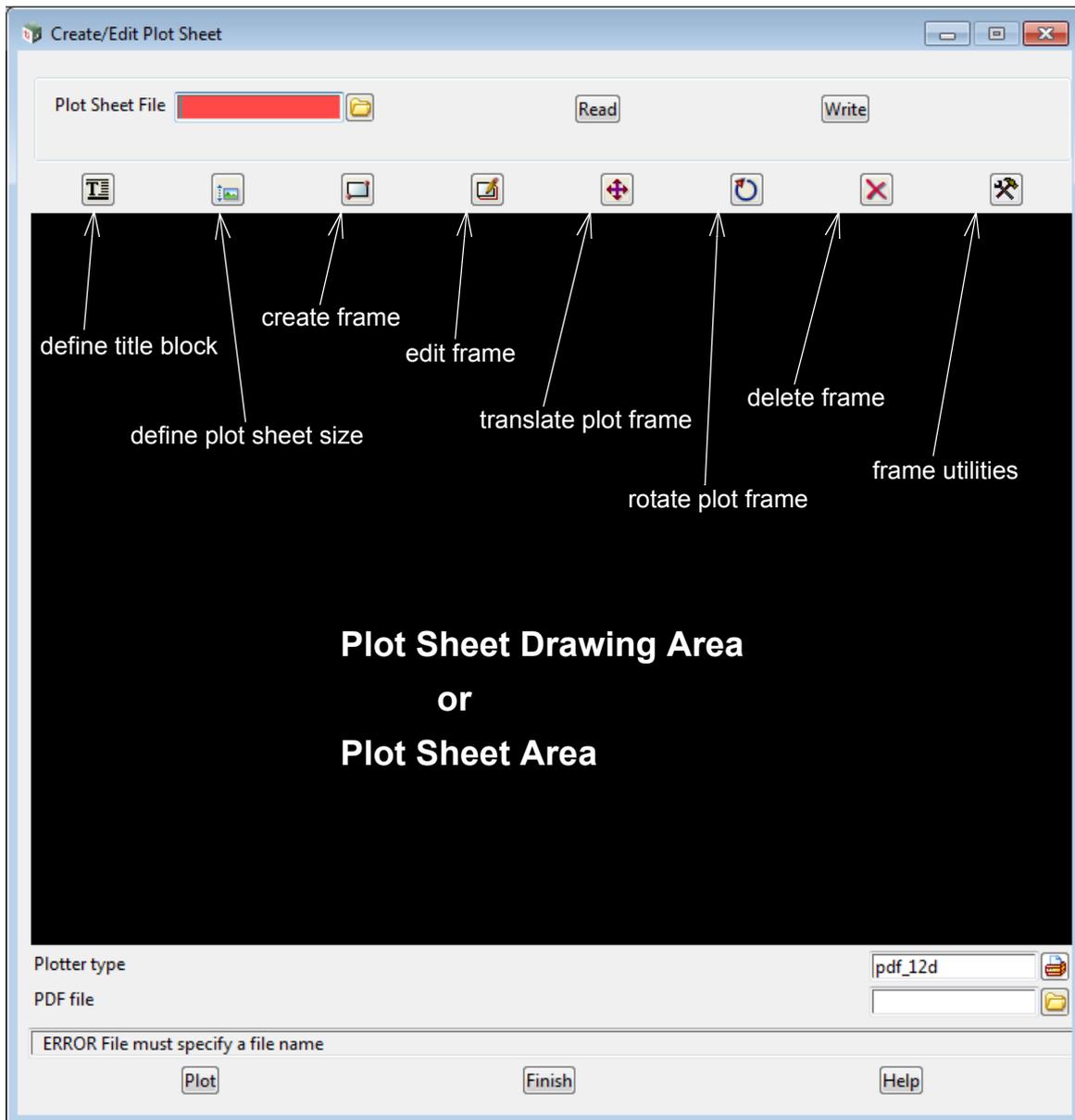
*when pressed, the existing plot sheet file with the name given in the **Plot sheet file** field is edited and the **Plot Sheet Create/Editor** panel is started with that file as the Plot Sheet File being edited.*

### 25.5.2.1 Plot Sheet Create/Edit

The **Plot Sheet Create/Edit** panel is for creating/editing the layout of plotting areas on a sheet of paper, and then to produce a pdf file of what is on the sheet of paper.

If the **Plot Sheet Create** option was used, then the **Plot Sheet Edit** panel is brought up with the new file loaded but there is nothing defined for the Plot Sheet.

If the **Plot Sheet Edit** option was used, then the **Plot Sheet Edit** panel is brought up with the existing file loaded so there will already be some information defined on the Plot Sheet.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Plot sheet file</b>	file box		*.12dpsf files
<i>name for the file to save/read in the information for a plot sheet layout.</i>			
<b>Read</b>	button		
<i>read in the plot sheet file given in the <b>Plot sheet file</b> field and load the information into the panel.</i>			

For any Frames in the file, a Plot Frame is created, and a model and the Plot Frame added to the model. If a view has been recorded for that model when the plot sheet file was written, and that view currently exists, then the model is added to that view.

**Write** button

write out the current information defined in the panel to a plot sheet file with the name given in the **Plot sheet file** field.

## Plot Sheet Area

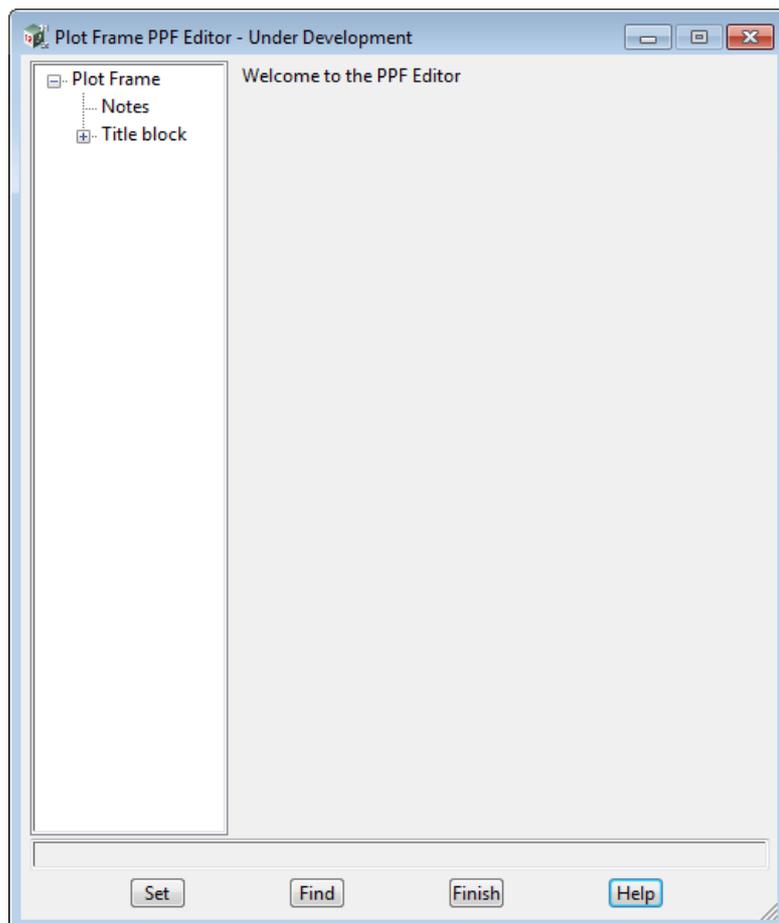
the black area represents the sheet of paper being plotted (the **Plot Sheet**) with (0,0) in the bottom left hand corner of the plot sheet.

The areas where plotting will occur on the Plot Sheet are defined by **Frames** and each created Frame is displayed in the black area. A Frame contains all the information for producing the plot inside the area of the Frame.

For example, a Frame may be a **Plan Frame** and defines an area of a Plan View to plot. The Frame then needs has a Scale to give the Frame a world size, a rotation and a World Coordinate to fully define a region to plot. Using these values, a Plot Frame is then created and linked to the Frame. The Plot Frame is added to a Plan View to define which data is to be plotted inside the Frame.

**Title block** button

brings up the Title Block section of a Plot Frame.

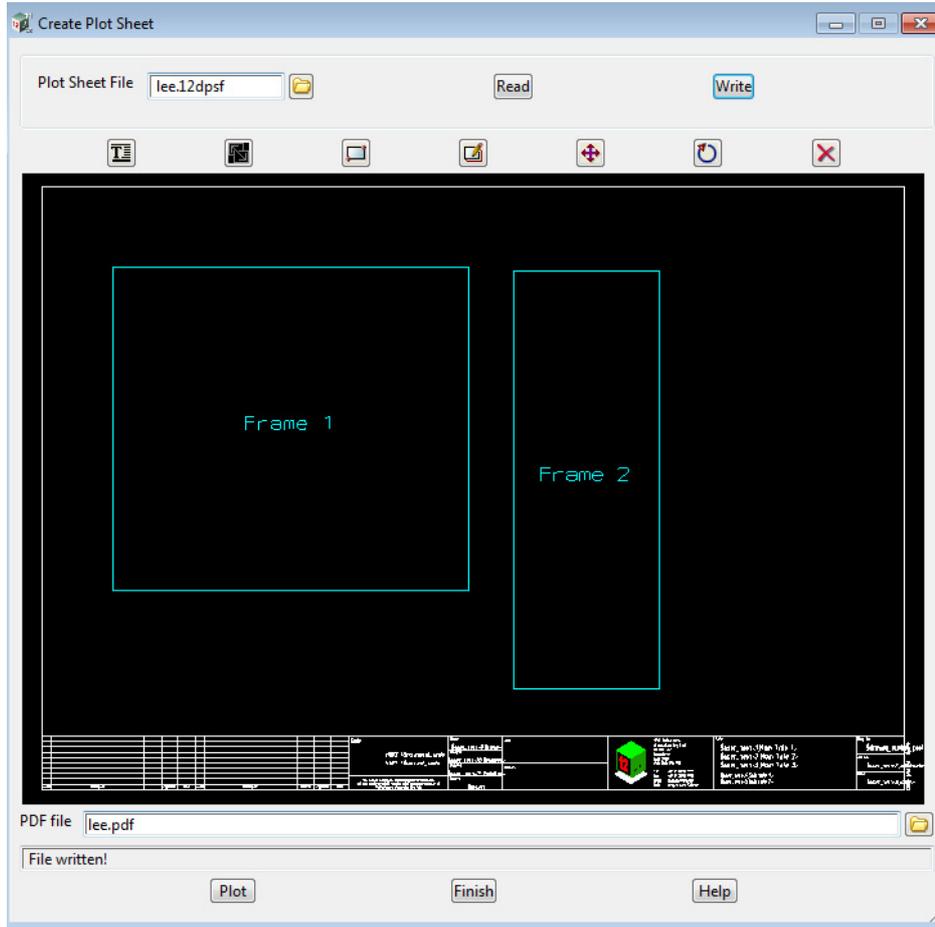


After the title block information has been filled in, the **Set** button must be pressed and then **Finish** to remove the panel.

If the title block file (tbj file) includes the Sheet size, then that size is automatically used to set the sheet size for the **Sheet Size** button.

If a user title block has been selected, the text and line work in the title block will be drawn in the Plot

Sheet Area so that you see where to add Frames to the Plot Sheet without overwriting the title block. No substitution is made for the \$ text variables in the title block file.



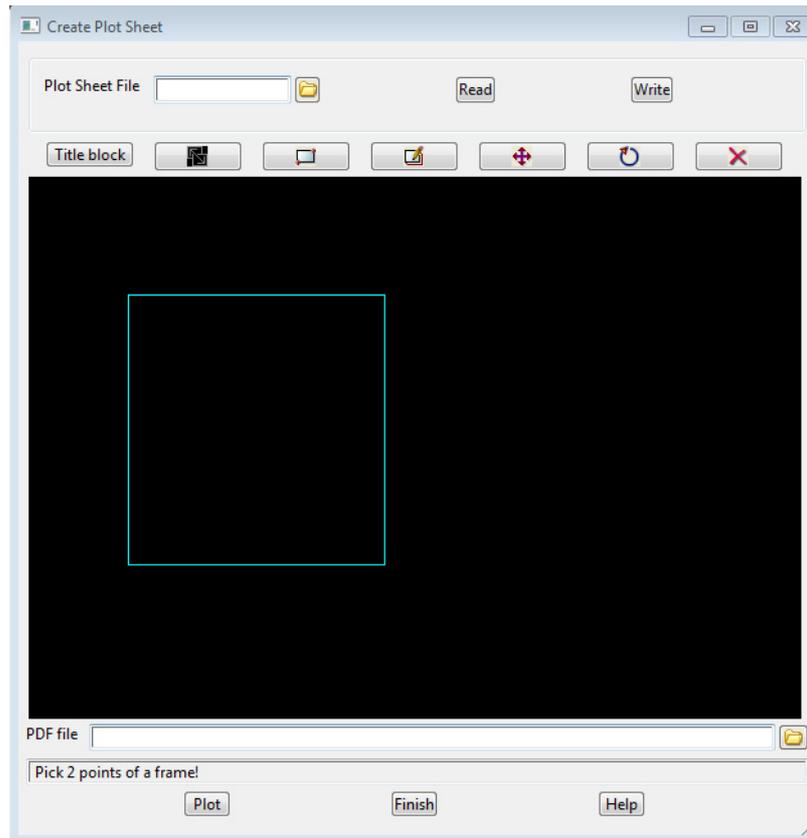
**Sheet size** button

brings up a **Sheet size** box for entering the size of the size of the Plot Sheet. Once a size has been typed in, or selected from the pop up, the Plot Sheet Area is drawn with an aspect ratio to match the selected paper size.

If there is a Sheet size in the selected user title block file, the Sheet size is automatically set to that size and the **Sheet Size** button does not need to be used.

**Create frame** button

this button is used to define a Frame in the plot sheet area. After clicking the button, two points are selected in the plot sheet area to define the area on the plot sheet that the Frame will plot to.



Currently all created Frames are Plan Frames so after the Frame is created, the **Create Frame for Plan Plot** panel is displayed. See [25.5.2.1.1 Create Frame for Plan Plot](#).

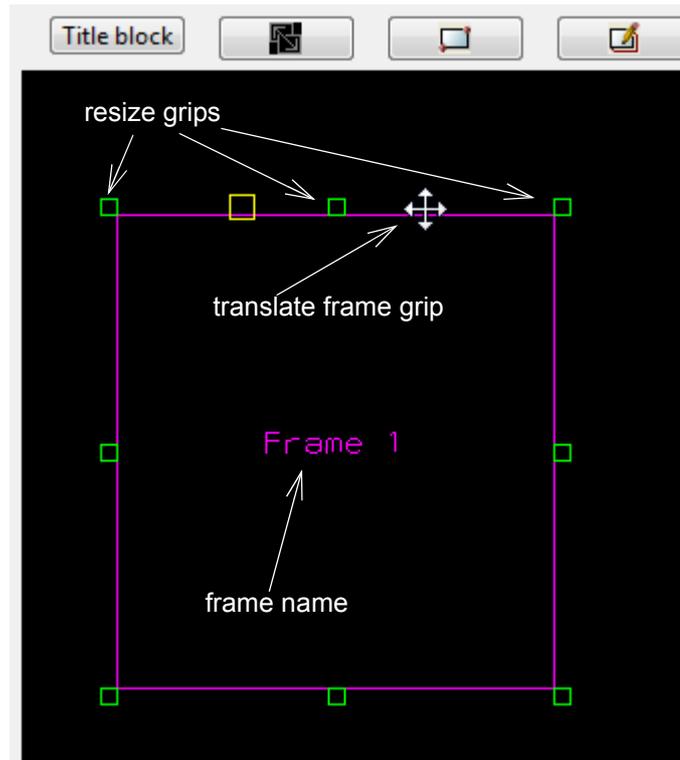
#### **Edit frame** button

*this button is used to edit an existing Frame in the plot sheet area.*

*After clicking the button, a Frame is selected by clicking LB **anywhere** inside the Frame in the Plot Sheet Area.*

*When the Frame has been tentatively picked with the LB but **not** yet **accepted**, **Resize** grips are displayed on the four corners and on the four sides of the Frame. These grips are used to resize the Frame, and hence the associated plotting area.*

*Plus when the cursor is over the edge of a Frame, a **Translate** grip is displayed. The **Translate** grip is used to move the Frame around in the Plot Sheet Area.*



How the **Resize** grips extend the plotting area depends on **Make centre of frame the origin**.

If **Make centre of frame the origin** is **ticked** then no matter which **resize grip** is used, the resulting plotting area is still **centred** on the **World Coordinate**.

If **Make centre of frame the origin** is **not** **ticked** then the plotting area is extended in the direction indicated by the **Resize grip** and the **World Coordinate** may be changed.

If you haven't accepted the **Frame** after the **Resizing** and **Translating**, then you can pick another **Frame** to **resize** or **translate** by simply clicking **LB** inside the new **Frame**.

When a **Frame** is accepted after a tentatively pick, then the **Create Frame for Plan Plot** panel is displayed so that the **Frame** properties can be modified. See [25.5.2.1.1 Create Frame for Plan Plot](#).

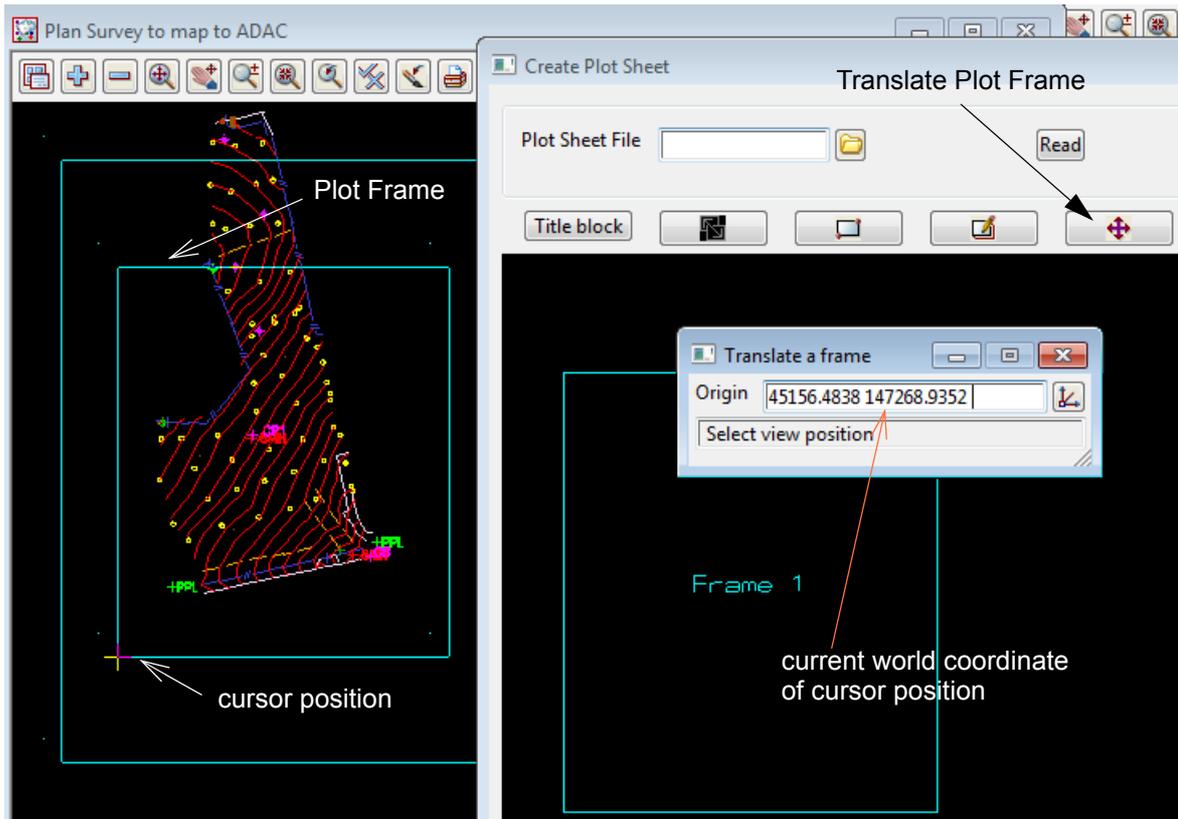
After the **Create Frame for Plan Plot** panel has been finished, the **Edit** is still active and another **Frame** can be selected to edit.

### Translate Plot Frame button

this button is used to change the value of the **World Coordinate** of a **Frame** by dynamically translating the **Plot Frame** that corresponds to the **Frame**.

After clicking the button, a **Frame** is selected by clicking **LB** **anywhere** inside the **Frame** in the **Plot Sheet Area** and then **accepting** the pick.

When the **Frame** is accepted, a **Translate a Frame** panel is displayed showing the value of the **World coordinate** for the **Frame**. Then when you move the cursor onto a **Plan View** which has the **Plot Frame** associated with the selected **Frame** on it, the **World Coordinate** of the **Plot Frame** and **Frame** are changed to the current cursor position. This continues until a new **World Coordinate** is selected on the **Plan View**.



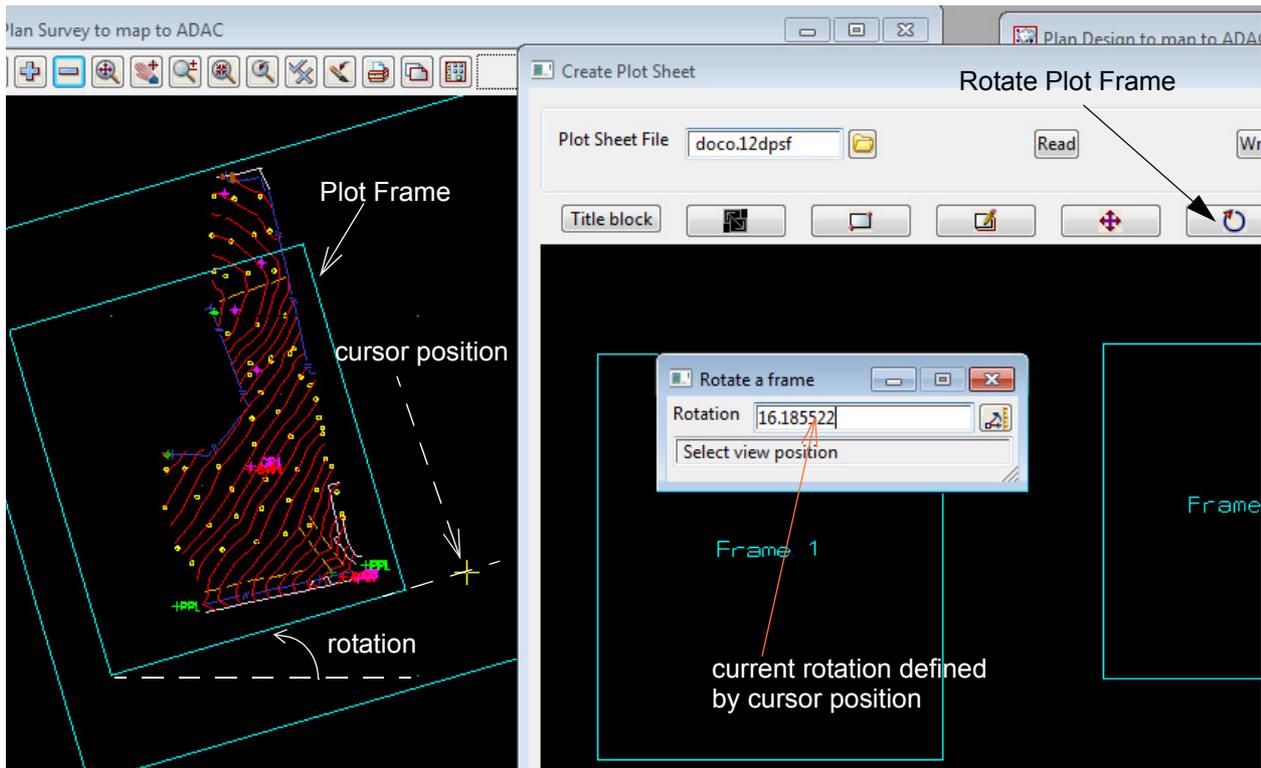
Once the Frame is translated then you can pick another Frame to translate.

**Rotate Plot Frame** button

this button is used to change the value of the **Rotation** of a Frame by dynamically rotating the **Plot Frame** that corresponds to the Frame.

After clicking the button, a Frame is selected by clicking LB **anywhere inside** the Frame in the Plot Sheet Area and then **accepting** the pick.

When the Frame is accepted, a **Rotate a Frame** panel is displayed showing the value of the **Rotation** for the Frame. Then when you move the cursor onto a Plan View which has the **Plot Frame** associated with the selected Frame on it, the **Rotation** of the Plot Frame and Frame are changed to the rotation defined by the current cursor position. This continues until a cursor position is accepted and that defines the new **Rotation** for the Plot Frame and Frame.



Once the Frame is rotated then you can pick another Frame to rotate.

**Delete Frame** button

this button is used to delete a Frame in the **Plot Sheet**. The associated Plot Frame and model containing the Plot Frame are also deleted.

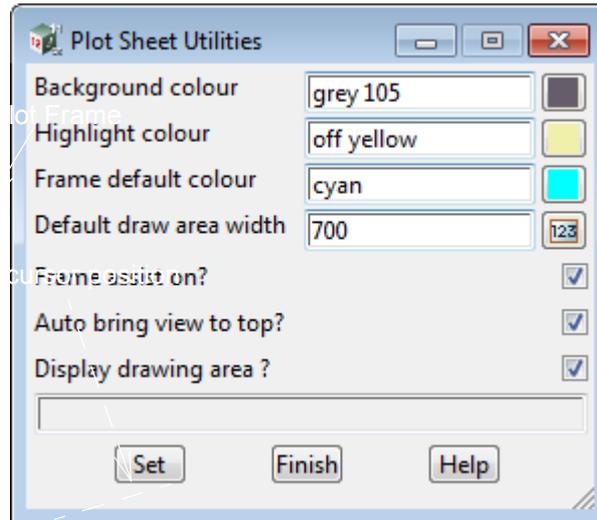
After clicking the button, a Frame is selected by clicking LB **anywhere inside** the Frame in the Plot Sheet Area and then accepting the pick.

When the Frame is accepted, the Frame is deleted, and its associated Plot Frame and the model containing the Plot Frame are also deleted.

Once a Frame is deleted, another Frame can then be selected to delete.

**Utilities** button

this button brings up the **Plot Sheet Utilities** panel to change the settings for the Plot Sheet.

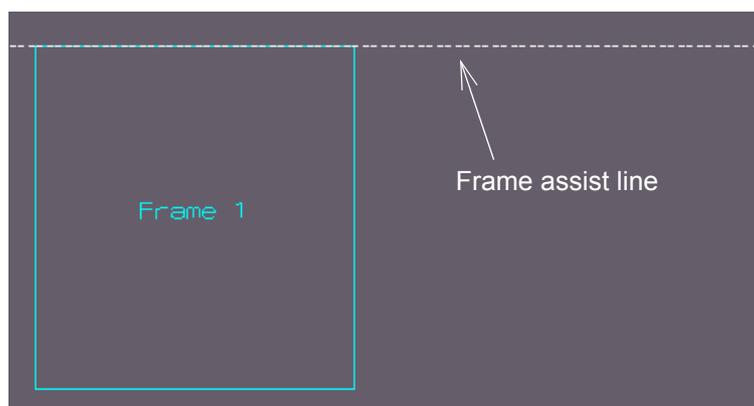


**Background colour**      colour box      grey 105      available colours  
*the colour for the background of the Plot Sheet drawing area. "Grey 105" is a good colour to use because both blank and white strings are visible when drawn on "Grey 105".*

**Highlight colour**      colour box      white      available colours  
*the colour to highlight the Frames in the Plot Sheet Drawing Area when they are selected.*

**Frame default colour**      colour box      cyan      available colours  
*the colour to draw the Frames borders in the Plot Sheet drawing area.*

**Frame assist on**      tick box      ticked  
*if **ticked**, when a new Frame is being created or an existing Frame moved in the Plot Sheet Area, when the cursor it is near the side of another Frame then guide lines are drawn in the Plot Sheet Area to help in placing the new Frame to be on the same horizontal or vertical line as the sides of existing frames.*



**Auto bring view to top?** tick box      ticked  
*if **ticked**, when the buttons Edit Frame or Delete Frame, or Plot Frame Move or Plot Frame Rotate, are selected and a Frame then selected, the View that the corresponding Plot Frame is on is brought to the top so that the View is not obscured by other Views.*

**Display drawing area**      tick box      ticked  
*if **ticked** and the title block file has a drawing area defined, it is shown in the Plot Sheet Area.*

**Fields and Buttons Under the Plot Sheet Area**

PDF file      file box      \*.pdf files

*name of the file to write the pdf plot out to.*

**Plot** button

*generate a pdf plot with the name given in the **PDF file** field.*

**Finish** button

*when the **Finish** button is selected, all the Plot Frames and the models containing the plot frames, that have been created by **Plot Sheet** are deleted.*

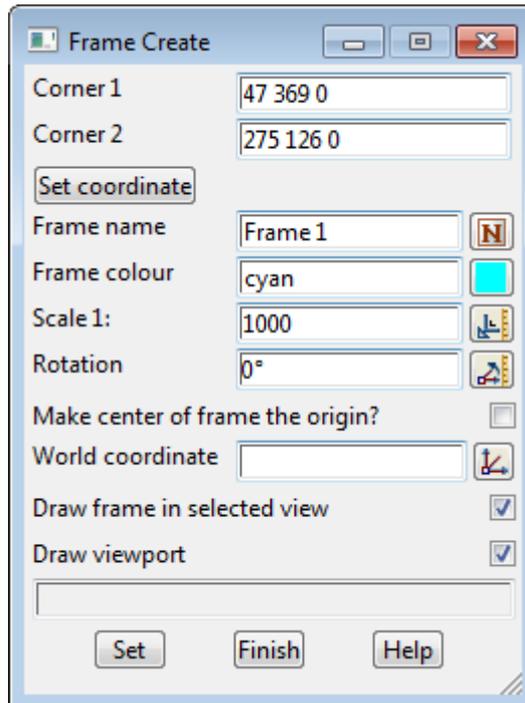
### 25.5.2.1.1 Create Frame for Plan Plot

The **Create Frame for Plan Plot** panel defines the information required to create a plan plot in the Frame.

The size of the Frame in **millimetres** is known from the Plot Sheet and so with a **Scale**, the Frame then defines a rectangle in world units (metres).

A **World Coordinate** for one point in the Frame and a **Rotation** then fully defines a world rectangle.

A **Plot Frame** of a given name can then be created and added to a view to represent the information to be plotted in the Frame.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Frame name</b>	text box		
<i>the name for the frame. It must be unique amongst all the Frames for the Plot Sheet. A name is automatically generated but it can be changed as long as it is still unique for the Plot Sheet.</i>			
<i>A <b>Plot Frame</b> will be created with the name of the plot sheet file (without the .12dpsf) followed by a space and the <b>Frame name</b>.</i>			
<i>A <b>model</b> will also be created with the name "<b>PF</b> " followed by the Plot Frame Name. The Plot Frame will be added to this model.</i>			
<b>Frame colour</b>	colour box		available colours
<i>the colour for the frame, and also for the Plot Frame that is created.</i>			
<b>Scale 1:</b>	measure box		available measures
<i>scale for the plan plot.</i>			
<b>Rotation</b>	measure box	0 degrees	available measures
<i>angle of rotation for the plot frame. The angle is in degrees and is measured in a counter clockwise direction from the positive x axis.</i>			
<b>Make centre of frame the origin?</b>	tick box		not ticked
<i>if <b>ticked</b>, the <b>World coordinate</b> is taken to be the world coordinate for the <b>centre</b> of the Frame.</i>			

If **not ticked**, the **World coordinate** is taken to be the world coordinate for the **bottom left hand corner** of the Frame.

**World coordinate**                      input    xyz ops menu

*x\_origin y\_origin z\_origin*  
this is a point in world units to be used as either the centre, or the bottom left hand corner, of the Frame. The values can either be typed in as three values separated by spaces, or selected by clicking LB on the **xyz ops** icon and then picking a point in a plan view.

When the **World coordinate** is selected from a plan view, then after the **Set** button is clicked, the plot frame is created, added to the created model, and the model added to the view that the **world coordinate** was selected from.

**Draw viewport**                      tick box                      ticked

if **ticked**, when a plot is generated, a box is drawn around the Frame.

**Set**    button

when **Set** is pressed, a Plot Frame of the name plot sheet file (without the .12dpsf) followed by a space and the **Frame name** is created. A model, with the name "PF " followed by the Plot Frame name, is then created and the Plot Frame is added to this model.

The model containing the Plot Frame is then added to the plan view that the **World coordinate** was selected from.

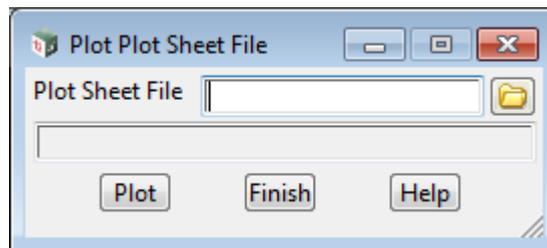
## 25.5.3 Plot Sheet Plot

**Current position on menu: Plot =>Plot sheet =>Plot**

**Plot sheet plot** is for plotting a plot sheet file that is on disk without having to bring up the plot sheet file in the **Plot Sheet Editor**.

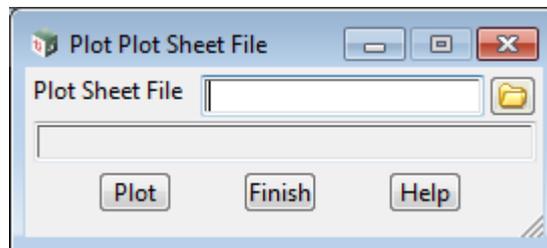
The **Plot** walk right operates in three ways.

1. The **Plot** walk-right displays a list of all the plot sheet files (\*.12dpsf files) and by double clicking on a plot sheet file in the list, the plot sheet file will be run and the associated plots produced.
2. If the walk right **Plot =>Plot sheet =>Plot** is being displayed from the **Main Menu** on the top of **12d Model**, then clicking on the name **Plot** on the **top of the walk right** will bring up the **Plot Plot Sheet File** panel.



Selecting the name of the plot sheet file in the field **Plot sheet file** and then pressing **Plot** will generate the plot.

3. If the walk right is being displayed from a pinned menu, then clicking on **Plot** on the **Plot Sheet menu without walking right** will also bring up the **Plot Plot Sheet File** panel.



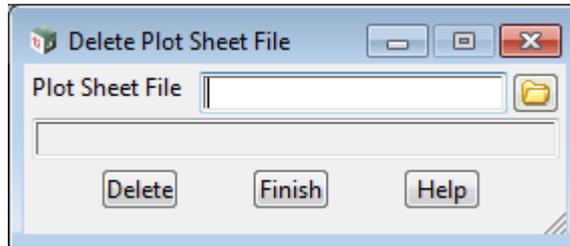
Again as in the previous case, selecting the name of the plot sheet file in the field **Plot sheet file** and then pressing **Plot** will generate the plot.

## 25.5.4 Plot Sheet Delete

**Current position on menu:** Plot =>Plot sheet =>Delete

**Plot sheet delete** is for deleting a plot sheet file from disk.

Selecting **Delete** brings up the **Delete Plot Sheet File** panel.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Plot sheet file</b> <i>name for the file to be deleted.</i>	file box		*.12dpsf files
<b>Delete</b> <i>delete the file given in the <b>Plot sheet file</b> field.</i>	button		

# 26. Utilities Menu

See

[26.1 Measure Bearing/Distance - Plan Area](#)

[26.2 DZ in Xfall Between 2 Strings Inquire Panel](#)

[26.3 Difference in Xfall Between 2 Strings Inquire \(Advanced\) Panel](#)

[26.4 Macro Prototypes File](#)

[26.5 Attribute Manipulator](#)

[26.6 Utilities =>A-G =>Explode](#)

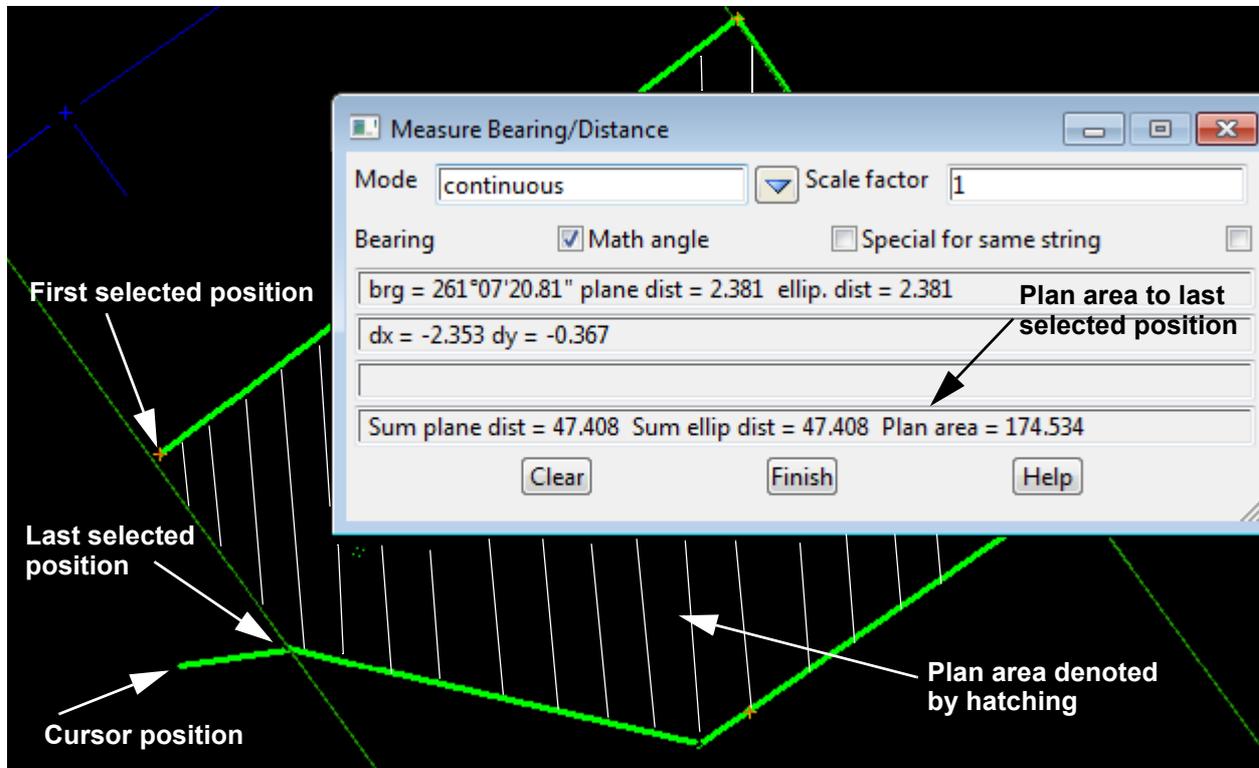
[26.7 Utilities =>H-Z =>Label Map](#)

## 26.1 Measure Bearing/Distance - Plan Area

For the **Measure/Bearing/Distance** panel brought up by clicking on

**Utilities =>Measure =>Bearing/Distance**

when in **continuous** mode, each time the next screen position is added to the highlighted figure, the **plan area** is calculated for the closed figure that is constructed by joining the first and last selected screen positions.

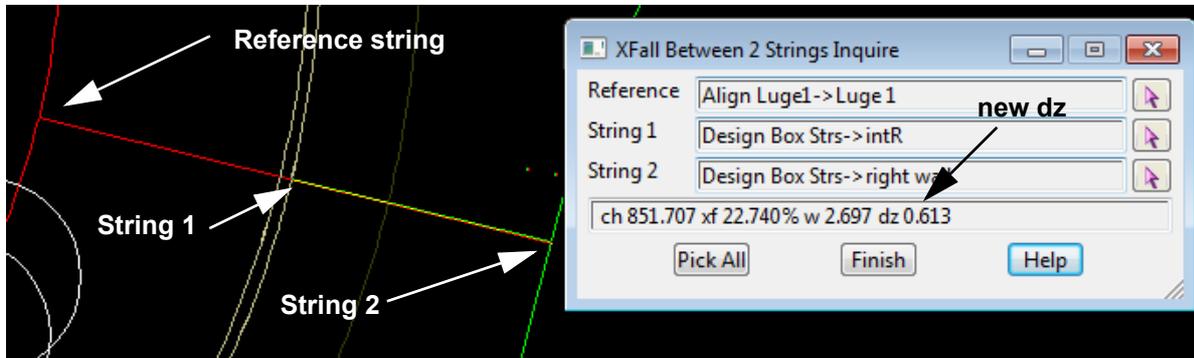


## 26.2 DZ in Xfall Between 2 Strings Inquire Panel

For the **Xfall Between 2 Strings Inquire** panel brought up by clicking on

**Utilities =>Measure =>XFall by strings**

the difference in the z values (**dz**) between the two strings is now displayed.

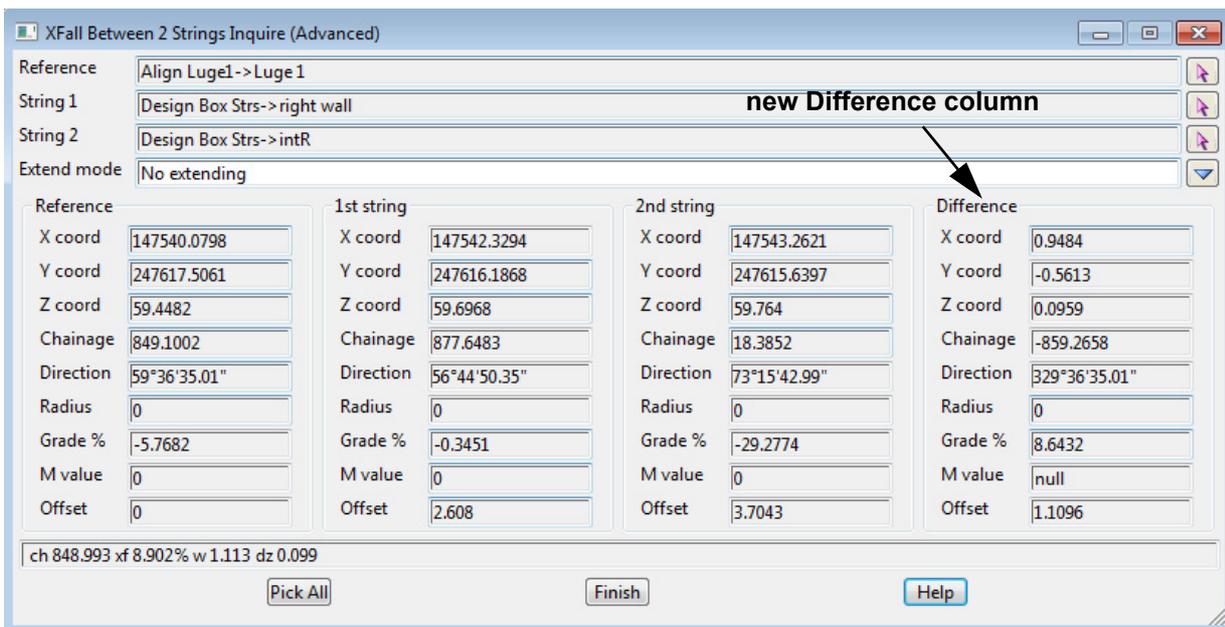


## 26.3 Difference in Xfall Between 2 Strings Inquire (Advanced) Panel

For the **Xfall Between 2 Strings Inquire (Advanced)** panel brought up by clicking on

**Utilities =>Measure =>XFall by strings (advanced)**

there is now a **Difference** column showing the differences for each of the values between the two strings.

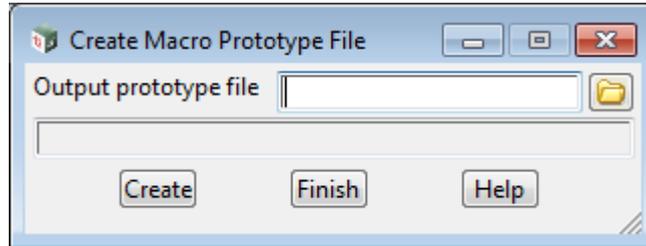


## 26.4 Macro Prototypes File

The option

**Utilities =>Macros =>Create prototype file**

creates a list of the prototypes of all the 12dPL function in the 12d Programming Language compiler that is available on your computer.



## 26.5 Attribute Manipulator

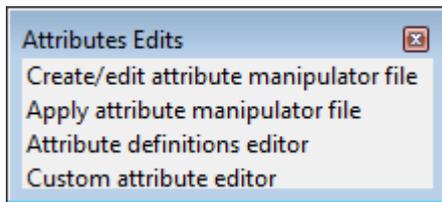
**Position of menu:** Utilities =>Attributes

This is a new option to manipulate attributes on a string.

Flat attribute structures can be made into tree structures and vice-versa, vertex and string attributes can be copied to string attributes and vice versa, attributes can be given default values and deleted. Some string properties such as name and colour can also be used to set attribute values and vice versa.

The attribute and property manipulations are stored in a **12d Attribute Manipulator (12dattmf)** file and this file is applied.

The two options are on the **Attributes** walk-right menu:



file to define attribute manipulations  
apply attribute manipulations

See

[26.5.1 Create/Edit Attribute Manipulator File](#)

[26.5.2 Apply Attribute Manipulator File](#)

## 26.5.1 Create/Edit Attribute Manipulator File

**Position of option on menu:** Utilities =>Attributes =>Create/Edit Attribute manipulator file

This option creates and edits a **12d Attribute Manipulator (12dattmf)** file which defines for any selected string, what string, vertex or segment user attributes are used to update other attributes on the string. Some string properties such as name and colour can also be used/modified.

The way that attributes are to be manipulated is defined as a list of rules in the file that are processed one after another in the order that they are listed in the file.

Each rule consists of two parts:

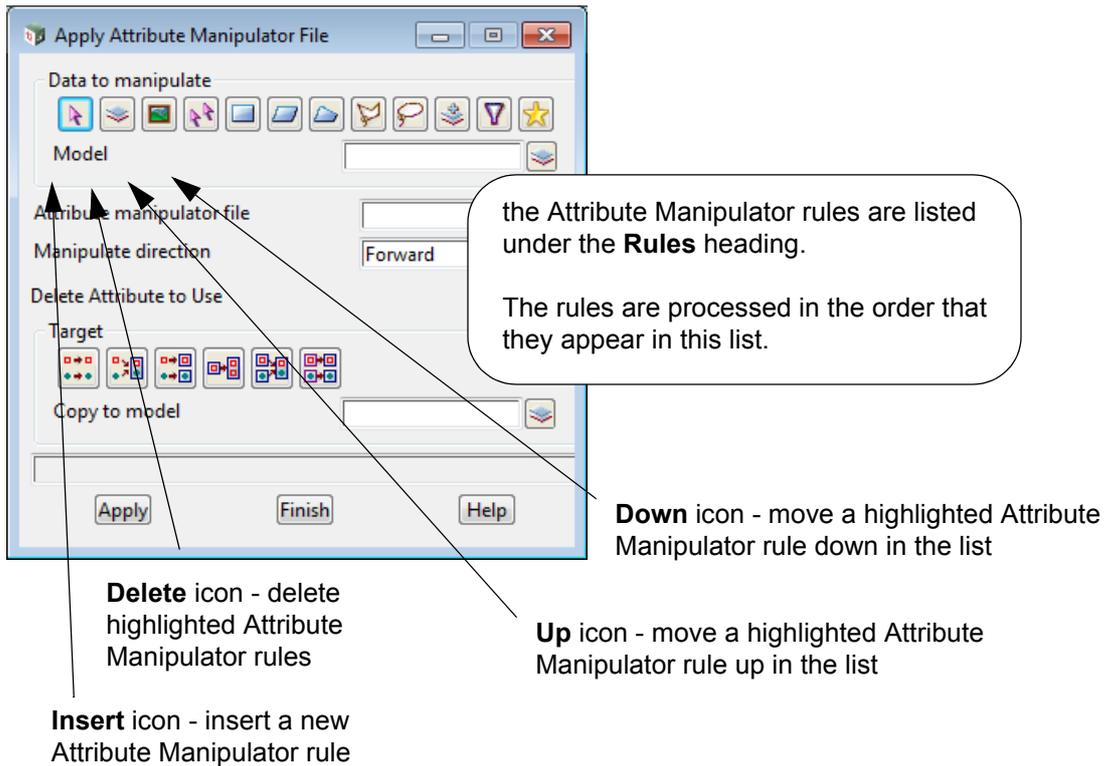
- (a) which attribute/property is used to take the values from - attribute/property to use
- (b) which attributes/property is to be modified - attributes/property to modify.

The **Apply Attribute Manipulator File** panel applies the **12dattmf** file to selected data (see [26.5.2 Apply Attribute Manipulator File](#)).

Although the documentation describes setting up and using the rules as in (a) and (b), it is possible to use the rule in reverse (backwards). That is, the information in (b) is used to modify the information in (a). So the rules can be used in a **Forward** or a **Backward** direction.

**Note:** For a string, the **Attribute Manipulator File** can also be used to set attributes to default values, delete attributes and also to set all the vertex/segment attributes of the same name to a value. This is useful when you don't know how many vertices there are in a string.

Selecting **Create/Edit Attributes Manipulator file** brings up the **Create/Edit Attribute Manipulator File** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>File</b>	file box		*.12dattribmf files

*name of the 12d Attribute Manipulator File to read or write.*

*The 12d Attribute Manipulator File is looked for as a standard **Library** file. See [41.3 Library, User Library, Customer Library](#).*

<b>Read</b>	button
-------------	--------

*click on this button and the file given in the **File** field is read in and loaded into the editor.*

<b>Write</b>	button
--------------	--------

*writes out the data in the editor to the file given in the **File** field.*

<b>Direction</b>	choice box	Forward, Backward, Both
------------------	------------	-------------------------

*each rule consists of an **Attribute to Use** section and a **Attribute to Modify** section. The names denote the fact that the information in the **Attribute to Use** section is used to modify the information in the **Attribute to Modify** section.*

*The standard configuration is what is displayed when **Direction** is **Forward**.*

*But the rules can be used the other way around. That is, the rules can be used **Backwards**.*

*So if **Direction** is **Backward**, the information shown in the **Attribute to Use** in the **Forward** direction, is displayed in the **Attribute to Modify** section, plus some extra field that are needed for **Attribute to***

**Modify.** And the information shown in the **Attribute to Modify** in the **Forward** direction, is displayed in the **Attribute to Use** section, minus the fields that are not needed in **Attribute to Use**.

When **Direction** is **Both**: the **Attribute to Use** section of the rule displays the fields that are displayed in the **Attribute to Use** section when the **Direction** is **Forward**, plus the extra fields that are required when the direction is reversed. The **Attribute to Modify** section displays the fields that are displayed in the **Attribute to Modify** section when the **Direction** is **Forward**.

So when **Direction** is **Both**, all the information for both direction is displayed at once but you need to be careful how you interpret it.

## Icons above Tree

For information on setting up attribute manipulator rules, see [26.5.1.1 Attribute Manipulator Rules](#)

### Insert



icon

inserts a new attribute manipulator rule. See [26.5.1.1 Attribute Manipulator Rules](#).)

### Delete

icon

deletes the highlighted attribute manipulator rules.

### Up

icon

move the highlighted attribute manipulator rule up in the list

### Down

icon

move the highlighted attribute manipulator rule down in the list.

## 12dattmf File Tree

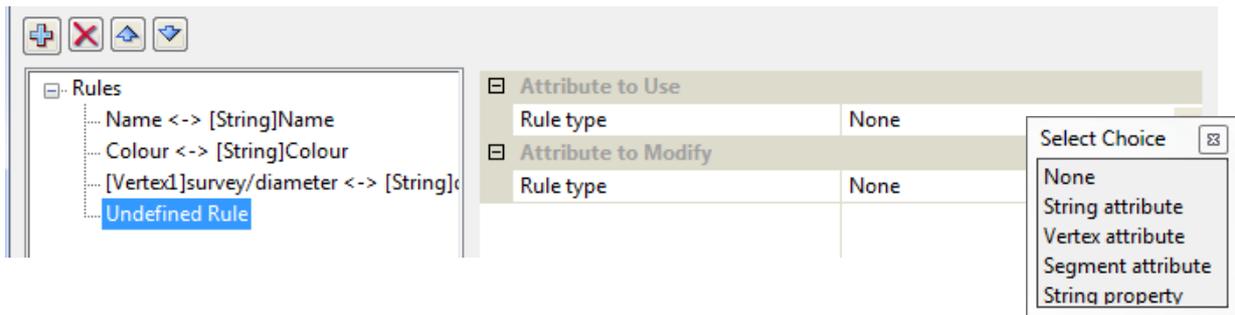
when an 12dattmf file is read in, it is displayed in the 12dattmf tree.

The definition of the Attribute Manipulator rules are given in the section [26.5.1.1 Attribute Manipulator Rules](#).

Continue to [26.5.1.1 Attribute Manipulator Rules](#), or the next major section [26.5.2 Apply Attribute Manipulator File](#) or return to [26.5 Attribute Manipulator](#).



The **Rule type** determines what extra information needs to be supplied for the rule.



**Using the Attributes to Use section and Attributes to Modify sections**

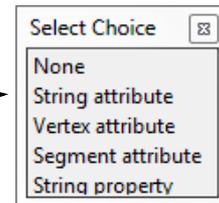
If **Rule type** in the section **Attribute to Use** is **None**, then the section **Attribute to Modify** is used to create an attribute with a default value OR give an existing attribute a default value OR DELETE an existing attribute.

If **Rule type** in the section **Attribute to Use** is **NOT None**, then the section **Attribute to Use** is used to select either a string property or a string, vertex or segment attribute, and its value is used to either create a new attribute or update an existing attribute or string property as specified in the **Attribute to Modify** section. The created/modified attribute can be forced to be a particular attribute type that does not have to be of the same type as the **Attribute to Use**. The created/modified attribute value can also be multiplied by a given factor.

**Fields in the section Attributes to Use**

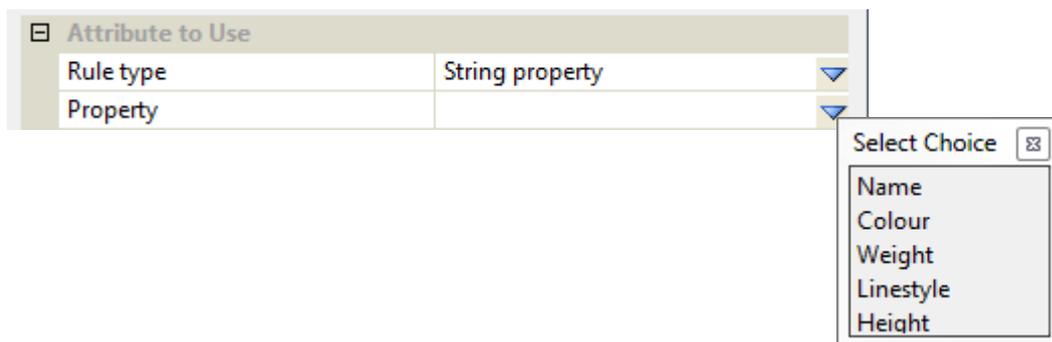
**Rule Type** choice box

choices for **Rule type** for the **Attribute to Use**



If **Rule type** is **None**, no other information is required. The section **Attribute to Modify** is then used to give an attribute a default value or to DELETE an attribute.

If **Rule type** is **String property**, the extra field **Property** is added to the panel.



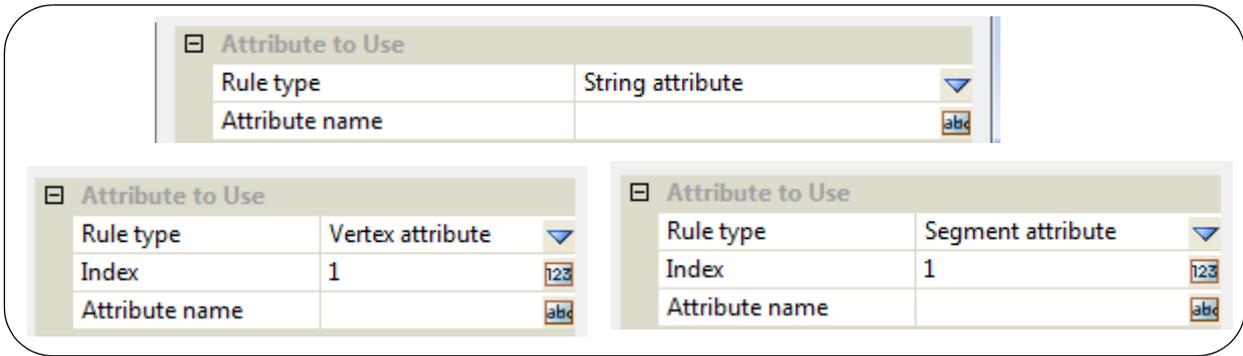
The string property to use is selected from the **Property** pop up

If **Rule type** is **String attribute**, **Vertex attribute** or **Segment attribute**, extra fields are added to uniquely specify the attribute to get a value from.

If **Rule type** is **String attribute**, the extra field **Attribute name** is added to the panel.

If **Rule type** is **Segment attribute** or **Vertex attribute**, the extra fields **Index** and **Attribute name** are

added to the panel.



**Index** integer box

**Index** is only for **Rule type** *Vertex attribute* and *Segment attribute*.

If **greater than zero**, the number of the vertex or the segment.

If **less than zero**, the absolute value of the number is the vertex/segment if the vertex/segment was numbered in the reverse order. That is, index -1 refers to the last vertex/segment in the string. -7 refers to the seventh last vertex/segment.

If **equal to zero**, the value for each vertex/segment **Attribute to Use** is used to set the value for the same number vertex/segment in the **Attribute to Modify**. In this case when **Index** is 0, the **Rule type** for the **Attribute to Modify** must be the same as the **Rule type** for **Attribute to Use** (*Vertex attribute* or *Segment attribute*) and also have **Index** equal to 0.

**Note:** the **Index** equal to 0 allows you to modify the attributes for every vertex/segment in a string.

**Attribute name** text box

the full path name of the string/vertex/segment attribute to use.

For example *Survey/diameter* where *Survey* is the group and *diameter* is the final name of the attribute.

**Note:** remember that attribute names and paths are case sensitive. That is *Lee* is different to *LEE*.

**Important Note on the section Attributes to Modify**

If **Rule type** in the section **Attribute to Use** is *None* then:

- (a) if the value in the **Default value** field in the section **Attribute to Modify** is **not blank** then the value of the attribute specified in the **Attribute to Modify** section is given the **Default value**.
- (b) if the value in the **Default value** field in the section **Attribute to Modify** is **blank** then the attribute specified in the **Attribute to Modify** section is **DELETED**.

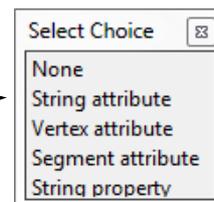
If **Rule type** in the section **Attribute to Use** is **NOT None**, then the section **Attribute to Modify** is used to create/modify the attribute or string property, specified in the **Attribute to Modify** section.

So the value to use to modify a string property, or create/modify a string, vertex or segment attribute will come from the **Attribute to Use** section OR be given by the **Default value** field in the **Attributes to Modify** section.

**Fields in the section Attributes to Modify**

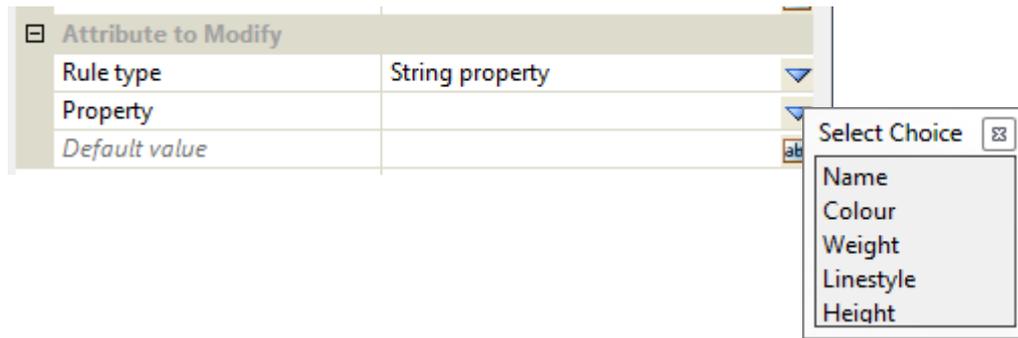
**Rule Type** choice box

choices for **Rule type** for the **Attribute to Modify**



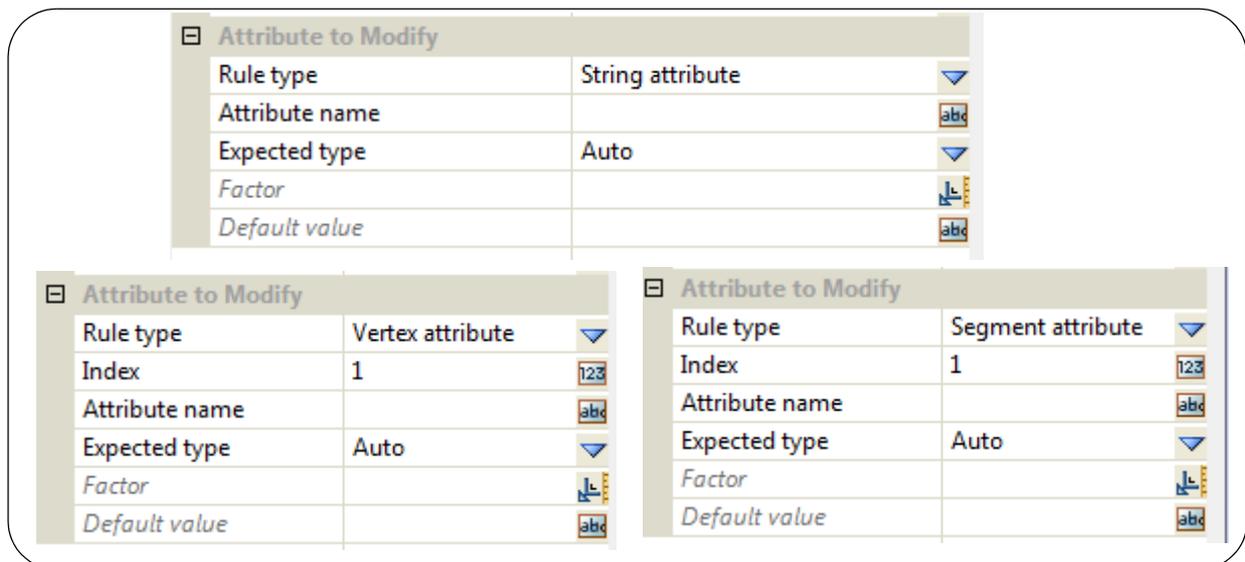
If **Rule type** is *None*, then the Rule has no meaning and is invalid.

If Rule type is **String property**, the extra fields **Property** and **Default value** are added to the panel.



- The string property selected in the **Property** field is updated by the value given by either
- (a) if the **Rule** in the **Attribute to Use** section was **NOT None** then the value in the **Attribute to Use** section is used for the update.
  - (b) if the **Rule** for the **Attribute to Use** section was **None**, then the value in the **Default value** field is used for the update.

If Rule type is **String attribute**, **Vertex attribute** or **Segment attribute**, the extra fields **Attribute name**, **Expected type**, **Factor** and **Default value** are added to the panel. For **Rule type Segment attribute** or **Vertex attribute**, the extra field **Index** is also added to the panel



The extra fields **Index** and **Attribute name** are needed to specify the attribute to set the value for:

**Index** integer box

only for **Rule type Vertex attribute** and **Segment attribute**.

If **greater than zero**, the number of the vertex or the segment of the **Attribute to Modify**.

If **less than zero**, the absolute value of the number is the vertex/segment of the **Attribute to Modify** if the vertex/segment was numbered in the reverse order. That is, index -1 refers to the last vertex/segment in the string, -7 refers to the seventh last vertex/segment.

If **equal to zero**, the value for each vertex/segment **Attribute to Use** is used to set the value for the same number vertex/segment in the **Attribute to Modify**. In this case when **Index** is 0, the **Rule type** for the **Attribute to Modify** must be the same as the **Rule type** for **Attribute to Use** (**Vertex attribute** or **Segment attribute**) and also have **Index** equal to 0.

**Note:** the **Index** equal to 0 allows you to modify the attributes for every vertex/segment in a string.

**Attribute name** text box

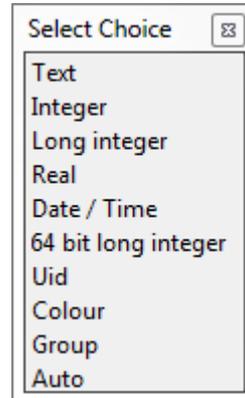
the full path name of the string/vertex/segment attribute to create/modify.

For example **Survey/diameter** where **Survey** is the group and **diameter** is the final name of the

attribute.

**Note:** remember that attribute names and paths are case sensitive. That is **Lee** is different to **LEE**.

**Expected type**                      choice box



If the **Attribute to Modify** doesn't exist and is being created then it will be of the **Expected type**. If **Auto** is selected then the new attribute will be of the same type as that of the **Attribute to Use**.

If the **Attribute to Modify** already exists and is not of the **Expected type**, then the existing attribute will be deleted and replaced by an attribute of the same name but of the **Expected type**. If **Auto** is selected then the new attribute will be of the same type as that of the **Attribute to Modify**. That is, the **Attribute to Use** is converted to the type of the **Attribute to Modify**.

**Factor**                                      real box

If **not blank**, the value of the attribute from the **Attribute to Use** section will be multiplied by the value of **Factor**.

If **blank**, **Factor** is not used.

For example, if the **Attribute to Use** is in metres and the new value is to be in millimetres then **Factor** is set to 1000.

**Default value**                              text box

The **Default value** is only used if the **Rule type** of the **Attribute to Use** is **None** or the attribute given by **Attribute to Use** does not exist.

If the **Rule type** of the **Attribute to Use** is **None** then:

If **Default value** is **not blank**, the **Attribute to Modify** is given this value.

If **Default value** is **blank**, the **Attribute to Modify** is **DELETED**.

If the **Attribute to Modify** is **Text** then a **Default value** of **\$null** will set the **Text Attribute to Modify** to blank. That is, no text.

If the **Attribute to Modify** is **Real** then a **Default value** of **\$null** will set the **Real Attribute to Modify** to the **12d Model** null value.

If the attribute of the **Attribute to Use** section **does not exist** then if the **Attribute to Modify** exists, it is given the **Default value**. If the **Attribute to Modify** doesn't exist then the rule is not used.

When the new rule is finished, press **Write** to update the file and save the rule. A name derived from information for the rule in the Forward direction is also automatically created and replaces the name **Undefined Rule**.

Continue to the next major section [26.5.2 Apply Attribute Manipulator File](#) or return to [26.5.1 Create/Edit Attribute Manipulator File](#) or [26.5 Attribute Manipulator](#) .

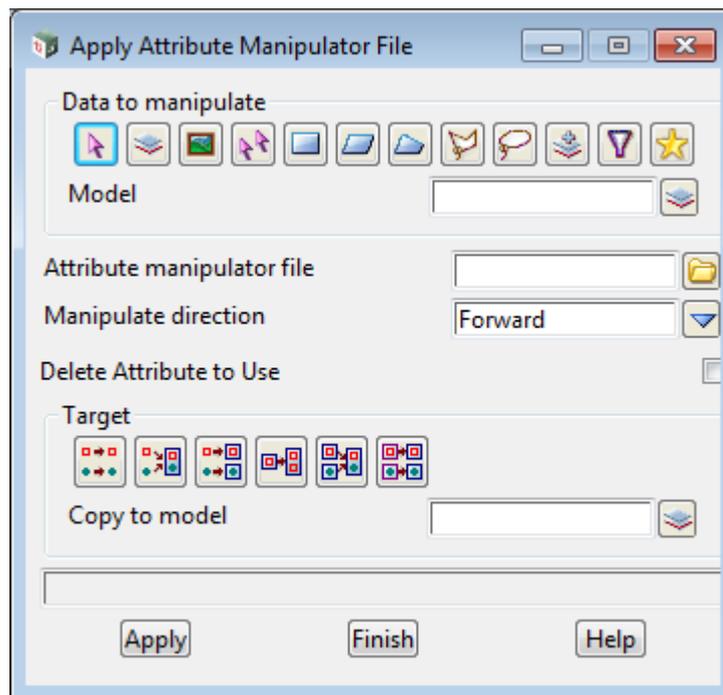
## 26.5.2 Apply Attribute Manipulator File

**Position of option on menu:** Utilities =>Attributes =>Apply attribute manipulator file

This option applies a **12dattmf** file to selected data to modify/update the user attributes or string properties with values from other User attributes on the strings. These may be string, vertex or segment attributes, and some string properties such as name and colour.

The **12dattmf** file is created and edited by the **Create/Edit Attribute Manipulator** panel (see [26.5.1 Create/Edit Attribute Manipulator File](#)).

Selecting **Apply attribute manipulator file** brings up the **Apply Attribute Manipulator File** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Data source type**

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to process. The selected strings will have user attributes updated by other user attributes etc as specified in a given 12dattribmf file.*

**Attribute manipulator file** file box

\*.12dattribmf files

*name of the 12d Attribute Manipulator file to use.*

**Manipulation direction**

choice box

Forward, Backward

*If Forward, the fields for the **Attribute to Use** section of the rule are those displayed when **Direction is Forward** in the **Create/Edit Attribute Manipulator** panel. The fields for the **Attribute to Modify** section of the rule are those displayed when **Direction is Forward** in the **Create/Edit Attribute Manipulator** panel.*

*If Backward, the fields for the **Attribute to Use** section of the rule are those displayed when **Direction is Backwards** in the **Create/Edit Attribute Manipulator** panel. The fields for the **Attribute to Modify** section of the rule are those displayed when **Direction is Backwards** in the **Create/Edit Attribute Manipulator** panel. See [26.5.1 Create/Edit Attribute Manipulator File](#).*

**Delete the Use attribute**    tick box

*if ticked, after each for each rule in the attribute manipulator file is run, the attribute referred to in the **Attribute to Use** section of the rule is deleted.  
If not ticked, no attributes are deleted.*

**Target type**

*Data target type - where to put the processed strings. For a full description go to [4.19.4 Data Target](#)*

**Target info**                    input

*extra information required for the target.*

**Apply**                            button

*when clicked, the selected data will have User attributes manipulated according to the selected 12dattribmf file.*

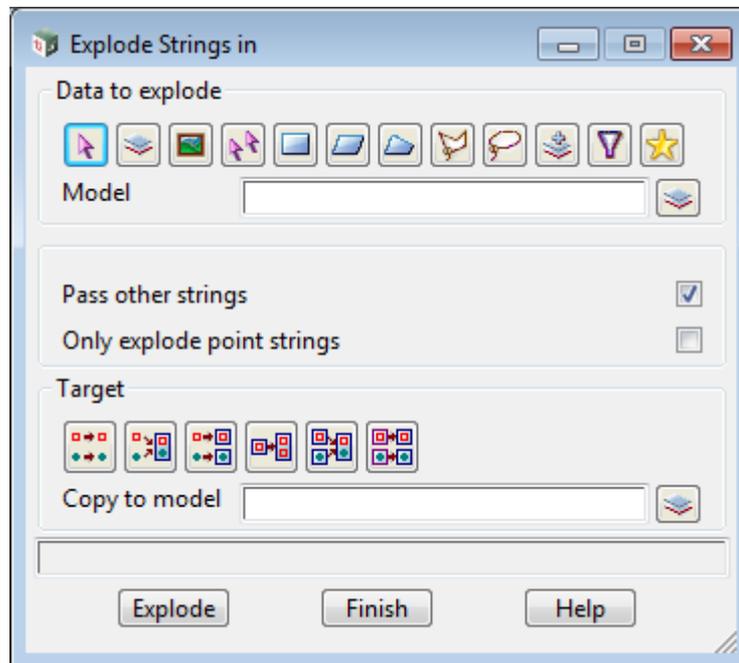
Return to [26.5 Attribute Manipulator](#) .

## 26.6 Utilities =>A-G =>Explode

The option

### Utilities =>A-G =>Explode

now has a tick box to allow only point strings, rather than point and line strings, to be exploded.



## 26.7 Utilities =>H-Z =>Label Map

The panel **Label Data by Label Map File** for applying a Label Map file was only under **File | O =>Label Map Files** has now also been added as the option

**Utilities =>H-Z =>Label Map**

# 27 ADAC

## ADAC - Asset Design and As Constructed

The **ADAC XML Schema** is a vendor independent XML format introduced to help solve the problems of collecting and exchanging Asset data.

The purpose of ADAC XML is to allow many different Authorities to work with one standardised file format. The format is fully defined and accessible to everyone.

The ADAC XML schema is controlled by the **Institute of Public Works Engineering Australasia (IPWEA)**. More information on ADAC at IPWEA can be found at [www.engicom.com.au/products/adac2/](http://www.engicom.com.au/products/adac2/)

See

[27.1 ADAC.XML Overview](#)

[27.2 12d Approach to ADAC XML](#)

[27.4 12d ADAC Menu](#)

## 27.1 ADAC.XML Overview

See

[27.1.1 ADAC XML Structure](#)

[27.1.2 ADAC Assets](#)

[27.1.3 ADAC Geometry Element](#)

[27.1.4 Guidelines from an Authority Requesting ADAC.XML](#)

### 27.1.1 ADAC XML Structure

The ADAC Schema is fully described by the *ADAC XSD (XML Schema Definition)*, which formally defines the structure and all the elements inside an ADAC XML file. That means that any ADAC.XML can be **validated** against the ADAC Schema XSD to see that it conforms to the Schema.

The **ADAC.XML Schema** and the ADAC.XSD are controlled and published by the [IPWEA](#).

ADAC Version 4.1.0 is the latest version and IPWEA publishes the *ADAC Schema Help Files* for Version 4.1 (and also the earlier 4.0) (see [www.engicom.com.au/products/adac2/](http://www.engicom.com.au/products/adac2/)).

The **ADAC XML Schema** is best thought of as a root or folder structure where the **Project** element encloses all data that is common to the whole project. So it is the root element or top folder.

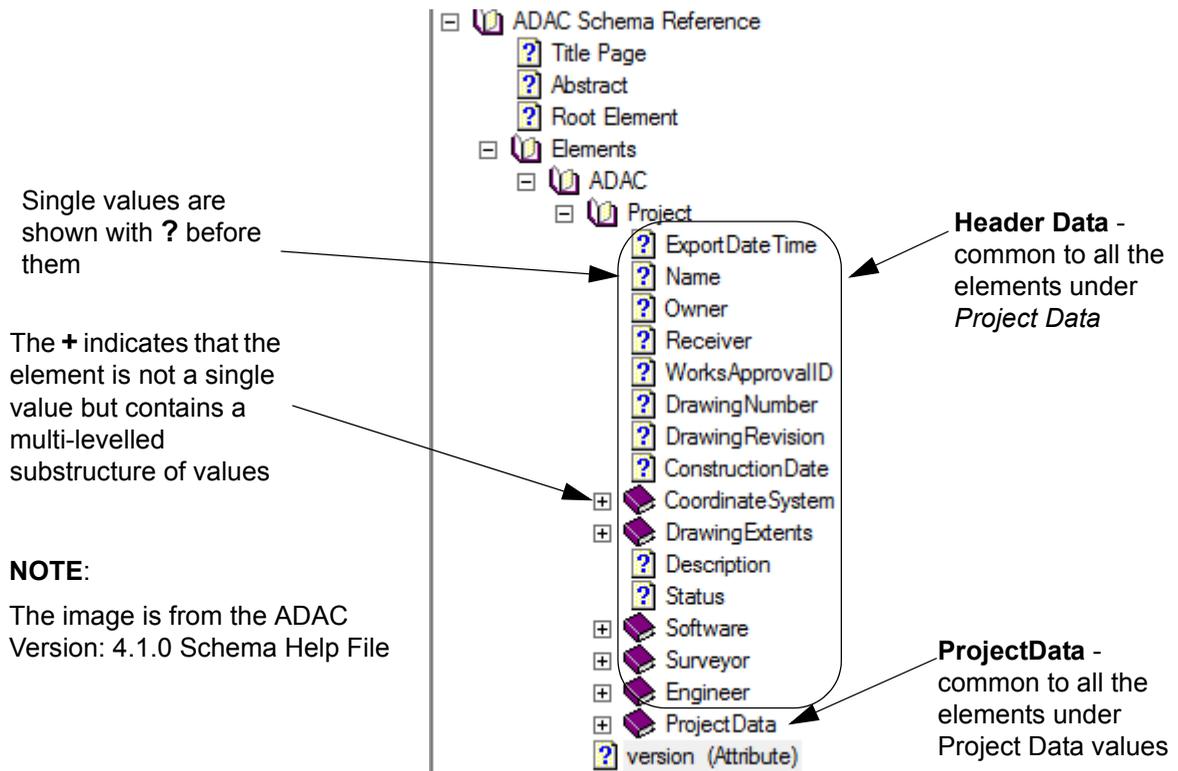
The ADAC standard currently says there can only be **one** Project in an ADAC.XML file.

Elements inside the **Project** element can be single values or multi-levelled structures (nodes) containing other single elements and substructures.

At the first level inside **Project**, the elements are:

Version, ExportDateTime, Name, Owner, Receiver, DrawingNumber, Drawing Revision, ConstructionDate, CoordinageSystem, Drawing Extents, Description, Status, Software, Surveyor, Engineer and ProjectData.

Some of these element are just single values (for example, *ExportDateTime* and *Owner*, and others, contain multi-levelled substructures of data (for example, *Surveyor*, *Engineer* and *ProjectData*).

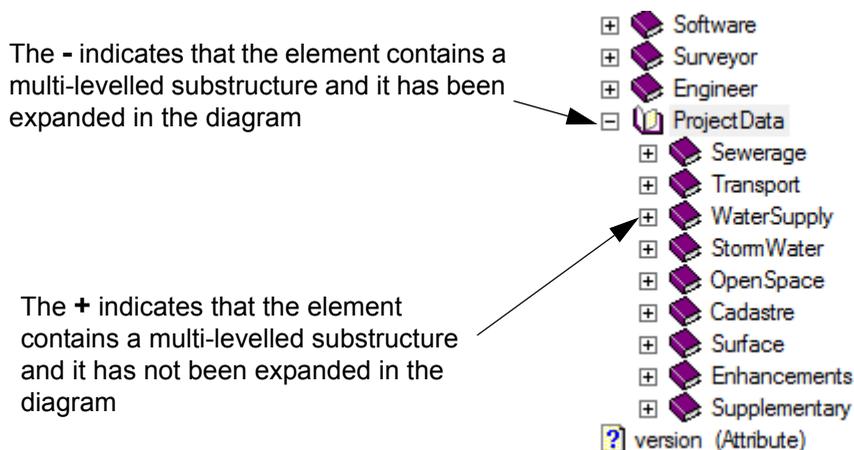


We will refer to all of these elements except **ProjectData** as **Header Data**.

The **Header Data** occurs only once and is at the beginning of the ADAC file.

The **ProjectData** element may appear insignificant in the expansion of the first level of **Project**, **BUT** it is the element that contains **all** the **physical items** covered by ADAC.

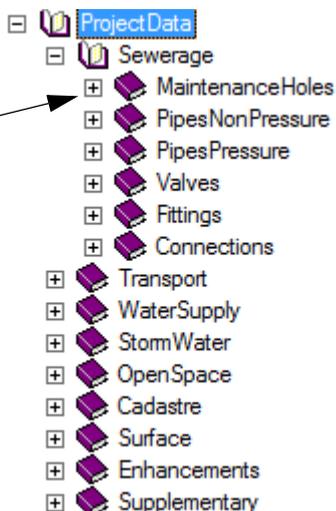
Expanding the **ProjectData** node shows it has the nine subelements *Sewerage, Transport, WaterSupply, OpenSpace, Cadastre, Surface, Enhancements* and *Supplementary*.



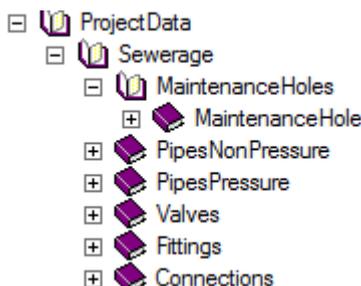
And each of the nine subelements have their own substructures

For example, **Sewerage** has the sub elements *MaintenanceHoles*, *PipesNonPressure*, *PipesPressure*, *Valves*, *Fittings* and *Connections*

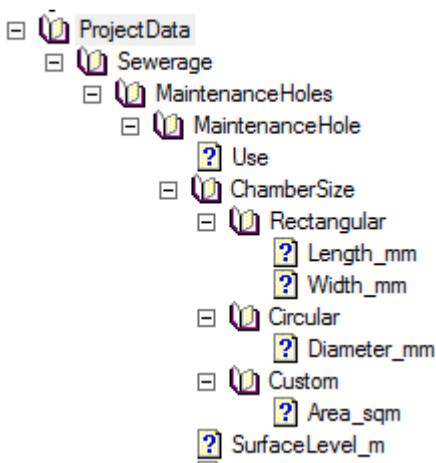
The + indicates that the element **MaintenanceHoles** contains a multi-levelled substructure



And **MaintenanceHoles** has a substructure that can be expanded out although it has only the one substructure, **MaintenanceHole**.



**MaintenanceHole** itself can be further expanded and part of **MaintenanceHole** is



All the structures and substructures are totally defined in the **ADAC XML Schema** and the structures must be strictly adhered to.

The **ADAC XML Schema** not only completely defines all the structures and substructures, but also has the **full definition** of all the elements, including the elements that can't be expanded any further (the **final** elements).

For the final elements, the ADAC Schema specifies everything. For example,

- their type - text, integer, real (float) or choices of these.
- what ranges the values may, or may not have
- if choices, then the list of possible choices (enumerations)
- if the item is optional (nillable = true) or mandatory (compulsory nillable = false)

For example, for the expansion of **MaintenanceHole**, the extract from the *ADAC XML 4.1.0 Schema* is

Independent Complex Data Type: **Feature\_Sewerage\_MaintenanceHole** [Back to Root Element](#)  
 Constrains: [objectModelSewerage\\_MaintenanceHoles\\_MaintenanceHole](#)

No documentation provided

Extends: [\[asset\\_abstract\]](#) (Contains all members of inherited class)  
 Sequence Occurrences: Minimum = 1 (Required) Maximum = 1

Sequence	Element	Annotations
	Element: <b>Use</b> { Data Type: <a href="#">sewer_mh_use</a> } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>Use or purpose of this MaintenanceHole in the network</i>	element is not optional description of the element
	Element: <b>ChamberSize</b> Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>Data structure describing the chamber configuration and dimensions.</i> Internal Complex Data Type Constraining: <a href="#">[ChamberSize]</a>	there is exactly one of them Choices for ChamberSize
	No documentation provided Choice Occurrences: Minimum = 1 (Required) Maximum = 1	Rectangular, Circular or Custom.
Choice	Element: <b>Rectangular</b> { Data Type: <a href="#">rectangular_struct_mm</a> } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>Data container for rectangular dimensions</i>	These all have a substructure. Links to the definitions of the substructures. For example Rectangular needs a Width and a Height
	Element: <b>Circular</b> { Data Type: <a href="#">circular_struct_mm</a> } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>Data container for circular dimensions</i>	
	Element: <b>Custom</b> { Data Type: <a href="#">custom_area_struct_sqm</a> } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>Custom Shaped chamber. Such a feature should be associated with a plan or document describing its layout</i>	
	Element: <b>SurfaceLevel_m</b> { Data Type: float } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>The height of the top surface of the lid, hatch, rim or roof. Surface level in meters against the vertical datum for this projec</i>	SurfaceLevel_m is a real number. It is a final element and fully defined
	Element: <b>InvertLevel_m</b> { Data Type: float } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>The height of the top surface of interior floor/bottom. Invert level in meters against the vertical datum for this project.</i>	
	Element: <b>FloorConstruction</b> { Data Type: <a href="#">construction_method</a> } Occurrences: Minimum = 1 (Required) Maximum = 1 Can be Nil = false <i>Method of chamber floor construction.</i>	

Hence this is how the **ADAC Schema** defines the **structure** of an ADAC.XML file and the definition of every element in the **ADAC Schema**.

If an XML file does not obey all the rules in the **ADAC Schema**, then the file is an **invalid XML file with respect to the ADAC Schema**.

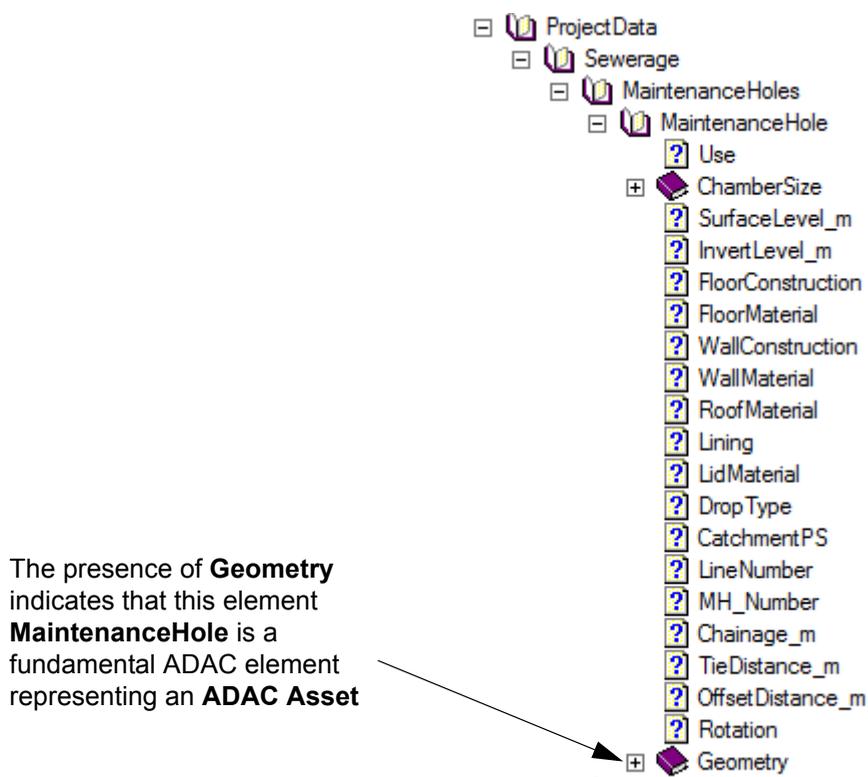
Continue to the next section [27.1.2 ADAC Assets](#) or return to [27.1 ADAC.XML Overview](#) or [27 ADAC](#).

## 27.1.2 ADAC Assets

Expanding substructures can go on at every level until no more expansions can occur.

But there is one important time to stop further expanding an element and that is when **one** of the substructures is **Geometry**.

The presence of **Geometry** indicates that the element is a **fundamental** ADAC element that represents an **ADAC Asset** or **ADAC Feature**.



So although **MaintenanceHole** has the substructure **ChamberSize** that can be further expanded, the presence of the **Geometry** element says that **MaintenanceHole** is a fundamental ADAC element representing an **ADAC Asset**.

For the ADAC 4.1.0 Schema, there are sixty-eight (68) such **ADAC Assets** including MaintenanceHole, PipeNonPressure, Pavement, RoadEdge, SubSoilDrain, Hydrant, RetainingWall, ElectricalConduit, Tree, Sign, Lot, Easement, SpotHeight and Contour.

Everything that is being recorded in an ADAC.XML file must be classified as one of the 68 ADAC Assets.

To further define what an ADAC Asset is, most of the ADAC Asset elements have a **Use** or **Type** which provides further information about that particular instance of an ADAC Asset.

For example, **MaintenanceHole** has a **Use** which **must exist** and must have one of the values (enumerations):

Independent Simple Data Type: [sewer\\_mh\\_use](#) [Back to Root Element](#)

Constrains: [Feature\\_Sewerage\\_MaintenanceHole.Use](#)

Allowed uses for a Maintenance Hole

Restriction of: [\[String\\_32\]](#)

Enumeration:	Overflow	<i>Overflow Maintenance Hole</i>
Enumeration:	Blank End	<i>Blank End</i>
Enumeration:	Pump Station	<i>Pit housing a sewer pump station</i>
Enumeration:	Valve Pit	<i>Access Pit for a Sewer Valve</i>
Enumeration:	Grit Collector MH	<i>Grit Collector Maintenance Hole</i>
Enumeration:	Outlet	<i>Outlet</i>
Enumeration:	Rising Main Discharge MH	<i>Rising Main Discharge Maintenance Hole</i>
Enumeration:	Vacuum Sewerage Pump Station	<i>Vacuum Sewerage Pump Station</i>
Enumeration:	Vacuum Sewerage MH	<i>Vacuum Sewer Maintenance Hole</i>
Enumeration:	Vacuum Lift	<i>Vacuum Lift</i>
Enumeration:	Storage Tank	<i>Storage Tank</i>
Enumeration:	Maintenance Hole	<i>Maintenance Hole</i>
Enumeration:	Maintenance Shaft	<i>Maintenance Shaft</i>

Continue to the next section [27.1.3 ADAC Geometry Element](#) or return to [27.1 ADAC.XML Overview](#) or [27 ADAC](#).

## 27.1.3 ADAC Geometry Element

Like everything in ADAC, the ADAC **Geometry** element is fully defined, but it is **different** for each of the **ADAC Assets**.

An **ADAC Geometry** can only be one of:

1. a **Point**. That is, an (x,y,z) coordinate.

For example, ADAC Assets with a Point Geometry include Sewer/MaintenanceHoles/ MaintenanceHole, WaterSupply/Hydrants/Hydrant, OpenSpace/Barbecues/Barbecue.

2. a **Polyline**. That is, a string made up of (x,y,z) points with straights and/or arcs between them (segments).

How many segments can be in the string, and whether arcs are allowed, can be different for each ADAC Asset with a Polyline Geometry. This can also vary with ADAC versions.

For example, in ADAC 4.0, Sewerage/PipesNonPressure/PipeNonPressure can only have one segment, and that segment must be a straight.

In ADAC 4.1, Transport/PipesNonPressure/PipeNonPressure can have any number of segments, and they can be straights or arcs.

3. a **Polygon**. That is, a closed polyline.

Whether arcs are allowed as segments of the polygon is specified for each ADAC Asset with a Polygon Geometry. This can also vary with ADAC versions.

ADAC Assets with a **Polygon Geometry** include *Transport/PavementAreas/Pavement* and *OpenSpace/LandscapeAreas/LandscapeArea*. Both of these can have any number of segments, and the segments can be straights or arcs.

If in an ADAC XML file, the Geometry written out for an ADAC Asset does not match the definition of the Geometry for that ADAC Asset, then it is breaking the rules and is an **invalid XML file with respect to the ADAC Schema**.

Continue to the next section [27.1.4 Guidelines from an Authority Requesting ADAC.XML](#) or return to [27.1 ADAC.XML Overview](#) or [27 ADAC](#).

## 27.1.4 Guidelines from an Authority Requesting ADAC.XML

Although ADAC XML has **sixty-eight** Assets, which extends to hundreds of different objects once **Use** and **Type** are considered, not every ADAC Asset is required by every Authority.

Even when an ADAC Asset is required by an Authority, not every element within that ADAC Asset may be wanted by that Authority.

Also for any ADAC Assets that are required for a particular Authority, what exactly does that **Geometry** represent?

For example, in the ADAC version 4.1.0 Schema, an OpenSpace/RetainingWalls/RetainingWall is defined as

"Represents a continuous retaining wall feature. Includes terrestrial, freshwater and marine revetment walls."

Independent Complex Data Type: **Feature\_OpenSpace\_RetainingWall** [Back to Root Element](#)

Constrains: [objectModelOpenSpace\\_RetainingWalls\\_RetainingWall](#)

*Represents a linear course of retaining wall.*

Extends: [\[asset\\_abstract\]](#) (Contains all members of inherited class)

Sequence Occurrences: Minimum = 1 (Required) Maximum = 1

Sequence

Element: **Use** { Data Type: [openspace\\_retainingwall\\_use](#) }

Occurrences: Minimum = 1 (Required) Maximum = 1

Can be Nil = false

*Context of use for this wall. i.e Terrestrial or Marine*

Use has three choices:  
Terrestrial, Freshwater  
and Marine

Element: **Material** { Data Type: [openspace\\_retainingwall\\_material](#) }

Occurrences: Minimum = 1 (Required) Maximum = 1

Can be Nil = false

*The material/type of Retaining Wall eg: Rock, Conc. Block, Conc. Crib*

Element: **Construction** { Data Type: [openspace\\_retainingwall\\_construction](#) }

Occurrences: Minimum = 1 (Required) Maximum = 1

Can be Nil = false

*Construction principle of this wall (eg: Gravity, Piled, Cantilever)*

Element: **Length\_m** { Data Type: [Float\\_Positive\\_NonZero](#) }

Occurrences: Minimum = 1 (Required) Maximum = 1

Can be Nil = false

*The lineal length of the wall in metres*

read number and must  
be greater than 0

Element: **Height\_m** { Data Type: [Float\\_Positive\\_NonZero](#) }

Occurrences: Minimum = 1 (Required) Maximum = 1

Can be Nil = false

*The height (or average height) of the wall in metres*

Geometry allows  
polylines with straights  
and arcs

Element: **Geometry** { Data Type: [geometry\\_linear\\_multipath\\_complex](#) }

Occurrences: Minimum = 1 (Required) Maximum = 1

Can be Nil = false

*The geometry representing this feature in coordinate space.*

And so the Geometry is a polyline with straights and arc segments.

But for, say, a *Terrestrial* RetainingWall, are the (x,y,z) coordinates of the Geometry the top of the retaining wall, the bottom of the retaining wall, or it doesn't matter?

So when asked to provide an ADAC XML file to an Authority, you need to ascertain from the Authority the following:

1. What ADAC Version is required. It will be either 4.0 or 4.1.
2. What ADAC Assets are required.

For example, Unity Water in Queensland is a Water Authority using ADAC 4.1 but they only require the ADAC Assets in *WaterSupply*.

3. What elements inside the required ADAC Assets are required.
4. What the geometry represents for the required ADAC Asset.

For example

## StormWater

From the ADAC.XML Guidelines from Bundaberg, Gladstone and Rockhampton Regional Councils

### EndStructure

Asset Capture: Simple point feature representing the top of the headwall.

Spatial Relationship: Headwall "floats" adjacent to the end of a StormWater pipe feature.



Figure 5

## Transport

From the ADAC.XML Guidelines from Moreton Bay Regional Council

- Roadways, including seals and pavement to be captured from “Nominal kerb line to Nominal kerb line” as a closed polyline as per “red-dashed” example pictured in figures 1 & 2 below. Note: Separate polygons will be required at changes in pavement and/or surfacing.
- Kerb line is captured on the nominal kerb line (invert of kerb and channel, face of kerb only) as shown by “yellow-dashed” line shown in figures 1 and 2 shown below.
- Sub-soil drains, where installed, are to be captured at kerb/seal junction as per the “blue-dashed” examples shown in figures 1 and 2.
- Road islands are captured as closed polyline/object from back of kerb. Individual sub-sections of traffic islands to be identified by different material types (i.e. paving, concrete, grassed) as per “green-dashed” line in figure 1.

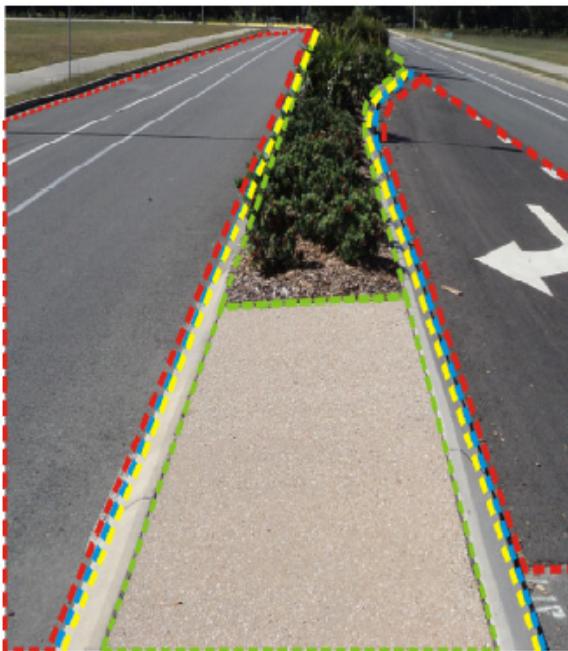


Figure 1

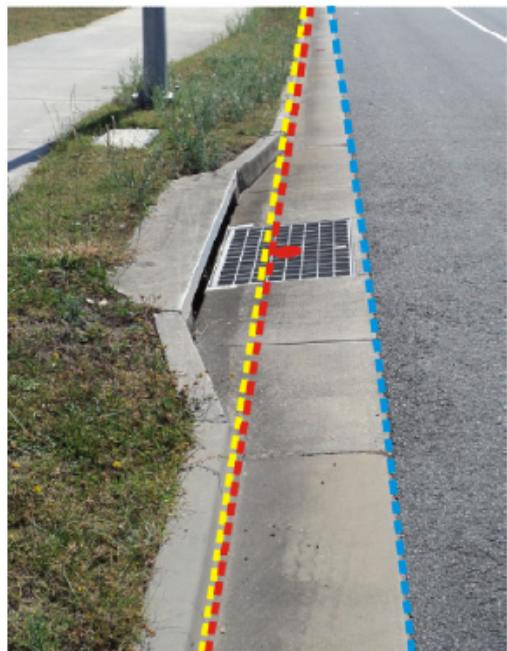


Figure 2

Return to [27.1 ADAC.XML Overview](#) or [27 ADAC](#).

## 27.2 12d Approach to ADAC XML

Creating an ADAC XML file could be approached as a spreadsheet type exercise where you type the values into a document, but this is little more than a manual data collection exercise, performed (usually as an extra expense) after the design or as-built surveys are completed.

This is **NOT** the **12d Model** approach.

The guiding principles to the **12d Model** approach are:

1. ADAC Data Fidelity

The ADAC Schema needs to be completely reflected inside **12d Model** so there is no ADAC data loss when writing and reading ADAC XML files.

2. 3D Engineering Data

All data is held as 3D Engineering data.

3. Round Tripping of Data

**12d Model** must be able to write out and read in ADAC data.

That is

**WRITE OUT** an ADAC XML file from data in **12d Model**

and

**READ IN** an ADAC XML file to **12d Model**.

4. Data Reuse

Data reuse must be maximised. That is, wherever possible, any required ADAC data that is already part of the **12d Model** data should NOT have to be re-entered for the ADAC XML.

So manual editing is to be minimised.

5. Integrated Work Flow

The creation/collection of the ADAC data must be **integrated** into the normal **12d Model** design-construct-as built workflow so that there is minimal extra work involved in producing ADAC XML files.

Continue to the next section [27.3 12d ADAC Workflow](#) or return to [27 ADAC](#).

## 27.3 12d ADAC Workflow

In **12d Model**, the **source** of the data to write out to an ADAC XML is usually from either a **design** or a **survey**.

1. For **Design** - the design has been created in **12d Model** and it is some of the data from the Design that is to be written out as an ADAC XML file.
2. For **Survey** - the assets have been constructed and surveyed, and the survey data has been read into **12d Model**. And it is some of the survey data that is to be written out as an ADAC XML file.

Although the type of data from a Design or a Survey can be totally different, the approach and steps inside **12d Model** for producing the ADAC XML file are very similar.

### Step 0. Before Doing any ADAC Projects - this is only done once, by one or two people

Before doing any 12d-ADAC processing, the first thing to do is set up the Company procedures that everyone will use for ADAC Design or ADAC Survey project. This is usually done once, by one or two people, **before** a company starts doing ADAC work with **12d Model**.

It involves:

- (a) deciding what **Data Preparation** is needed for your company's survey or design before it is ready to produce an ADAC XML file.
- (b) setting up the Company **User ADAC** menu.
- (c) to allow for fast **bulk** processing, set up an ADAC **12d Map File for Design** and/or an ADAC **12d Map File for Survey** to assign the ADAC Asset type to a string from design or survey.

This is not essential and can be done, or added to, as you get more ADAC experience. But using an ADAC Map File is where the big benefits come from on most projects.

For information on each of these activities, see [27.6 Setting Up for ADAC](#).

### Step 1. Setting Up a New ADAC Project - this is done once for each new ADAC project

For a new **12d Model** project, the first step is to set it up as either an ADAC Design or an ADAC Survey project.

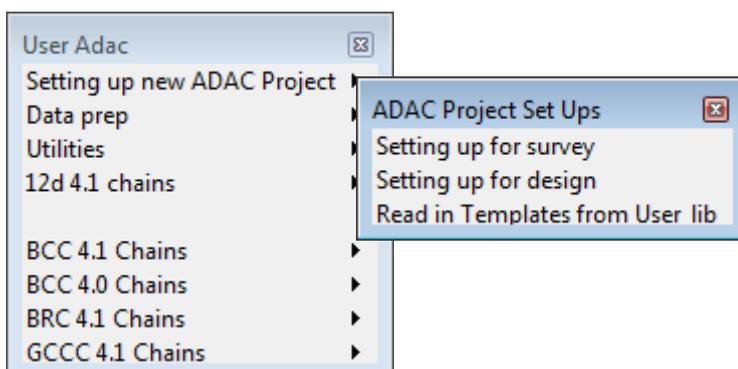
This is done **once for a new project** and only involves clicking a menu item:

**File I/O =>ADAC =>User =>Setting up for survey** - for surveys

For more information on what this option does, see [27.4.9.1.1 Setting Up for Survey](#).

**File I/O =>ADAC =>User =>Setting up for design** - for designers.

For more information on what this option does, see [27.4.9.1.2 Setting Up for Design](#).



**Step 2. Creating the ADAC Header Data - only done once for each ADAC project.**

The *ADAC Header Data* is only entered once for a project. This requires only a small amount of information such as the ADAC project name, the Owner and Receiver, the coordinate system used and (optionally) the Surveyor and the Engineer. See [27.4.1 Create Header](#).

If a *Header template* is used then most of the Header Data is automatically loaded.

**Step 3. Preparing the Data to go to ADAC - done as required in each ADAC project**

The design or survey data may need some preparation before it can go out to an ADAC XML file.

There is a variety of tools provided in **12d Model** ranging from setting Lot numbers, Plan Numbers and Areas, to splitting up a road edge into separate kerb types as required by ADAC.

See [27.4.9.2 Data Prep](#)

**Step 4. For Some Items, Assigning the ADAC Asset Type by Hand**

Some data may be selected by hand to assign its ADAC Asset type using the **Create ADAC Asset** option. See [27.4.2 Create ADAC Asset](#)

As your **ADAC Map Files** are created and extending, less data will need to be selected by hand. Instead the ADAC Asset type it will be automatically assigned.

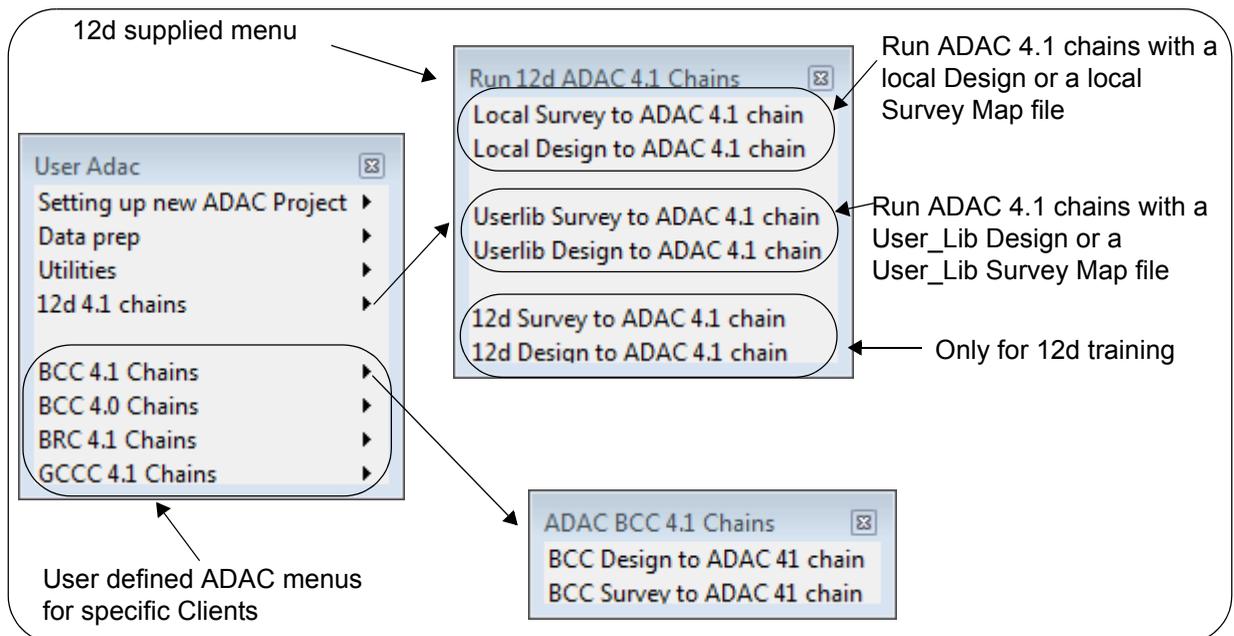
**Step 5. Running a Design or Survey Chain - run regularly in an ADAC project**

All the hard work for ADAC is done by either running an ADAC Design Chain or an ADAC Survey Chain. And to do that only involves clicking a menu item such:

**File I/O =>ADAC =>User =>Appropriate Client Survey chain** - for surveys

**File I/O =>ADAC =>User =>Appropriate Client Design chain** - for designers.

For more information on these chains, see [27.5 ADAC Design and Survey Chains](#).



In summary, what the *ADAC Design* or *ADAC Survey* chain does is:

(a) Uses the ADAC Map File to Assign the ADAC Asset Type

Any data going out to ADAC must be one of the ADAC Assets.

So either the ADAC Asset type has been assigned by hand, or the assignment is automatically done using an **ADAC Map File** that has already been set up.

- (b) Sets ADAC values from the 12d String Geometry and User Attributes.

Much of the required ADAC data is already contained within the 12d strings or in User Attributes.

For example, if the Drainage or Sewer was designed with **12d Model**, then much of the ADAC data can come directly from the drainage and sewer strings. For example, maintenance holes and chambers, depths, and pipe dimensions come directly from the drainage string, and any required 2D and 3D lengths are calculated from the drainage string.

User attributes picked up in the field by the surveyors, or added in the **Data Prep** step, are also loaded into the specified ADAC attributes.

- (c) Creates the ADAC Geometry.

For each ADAC Asset, the ADAC Geometry is automatically generated from the 12d string geometry.

For example, if the ADAC Asset has Polyline Geometry, the (x,y,z) coordinates for each vertex of the string are loaded into the ADAC Geometry.

#### **Step 6. Validating the Data, Generating Reports, Generating the ADAC Geometry - done as required**

A Validation option checks the data against the ADAC Schema, and any errors are flagged and are easily found using **12d Model's** Intelligent logs lines. This is a simple process and is done by running only one panel. See [27.4.4 Validate](#).

A report of the ADAC Data can be produced in HTML, Text or PDF format. This could be for your records or used as another checking tool. This is a simple process and is done by running only one panel. See [27.4.5 Report](#).

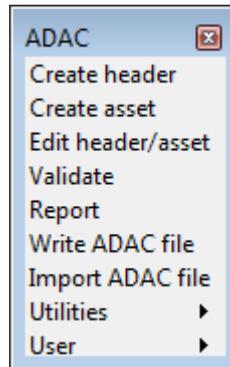
#### **Step 7. Writing the Data out to ADAC XML - done as required**

This is a simple process and is done by running just the one panel. See [27.4.6 Write ADAC XML File](#).

Continue to the next section [27.4 12d ADAC Menu](#) or return to [27 ADAC](#).

## 27.4 12d ADAC Menu

Position of menu: File I/O =>ADAC



See

[27.4.1 Create Header](#)

[27.4.2 Create ADAC Asset](#)

[27.4.3 Edit ADAC Header/Asset](#)

[27.4.4 Validate](#)

[27.4.5 Report](#)

[27.4.6 Write ADAC XML File](#)

[27.4.7 Import ADAC XML File](#)

[27.4.8 ADAC Utilities](#)

[27.4.9 User ADAC](#)

## 27.4.1 Create Header

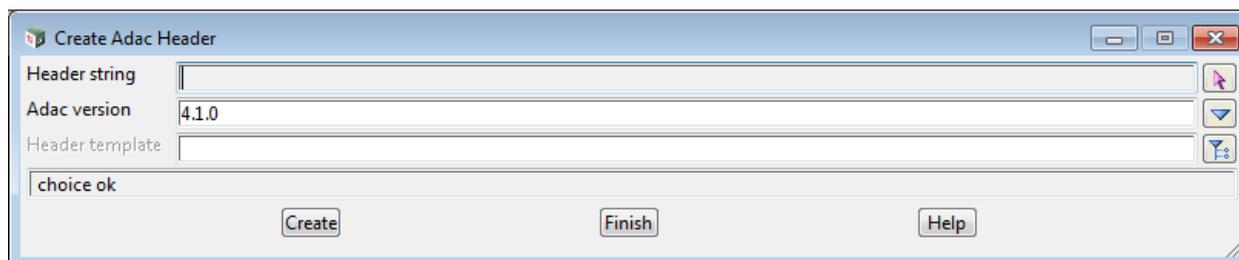
**Position of option on menu:** File I/O =>ADAC =>Create header

To produce an ADAC XML file, **12d Model** needs exactly one string with the ADAC Header data, and zero or more strings representing ADAC Asset data.

The **Create header** option takes a one vertex string, and creates as string attributes, the attributes needed to make it an ADAC Header. A user selected ADAC Header Template can be used to set the values for any of the ADAC attributes in the ADAC Header.

The **ADAC Header Editor** is then brought up so any of the created ADAC Header attributes can be edited.

Selecting Create header brings up the **Create ADAC Header** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Header string</b> <i>select a one vertex string to set up with the ADAC Header Data attributes.</i>	string select		
<b>ADAC version</b> <i>the version of ADAC to use when creating the ADAC Header data. The default value is set by the environment variable ADAC_VERSION_4D which is set in the <b>Edit Environment Variables</b> panel at <b>External Apps &gt;ADAC</b>. See <a href="#">8.6.3 env.4d</a>.</i>	choice box	from env.4d	4.0.0, 4.1.0
<b>Header template</b> <i>the name of an ADAC Header template that is used to fill in some of the ADAC Header information for the Header string.</i>	ADAC Header box		available ADAC Header templates
<b>Create</b> <i>clicking <b>Create</b> creates an ADAC Header structure as 12d ADAC attributes for the string, and if there is a <b>Header template</b>, it loads the string's ADAC attributes with values from the Header Template. The <a href="#">27.4.2 Create ADAC Asset</a> panel is then brought up so that the ADAC Header data can be create/edited.</i>	button		

Continue to the next section [27.4.2 Create ADAC Asset](#) or return to [27.4 12d ADAC Menu](#) or [27 ADAC](#).

## 27.4.2 Create ADAC Asset

**Position of option on menu:** File I/O =>ADAC =>Create Asset

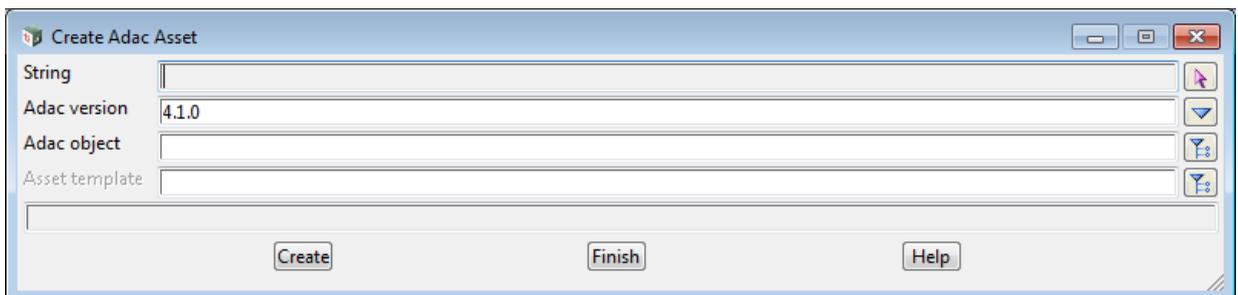
To produce an ADAC XML file, **12d Model** needs one string with the ADAC Header data, and zero or more strings with ADAC Asset data.

The **Create ADAC Asset** option takes a string with the appropriate geometry and creates as string attributes the ADAC Asset attributes for the user selected ADAC Asset type. This is referred to as **assigning the ADAC Asset type**, or **assigning the ADAC type**, to the string.

Also, a user selected ADAC Asset Template can be used to set the values for any of the ADAC attributes.

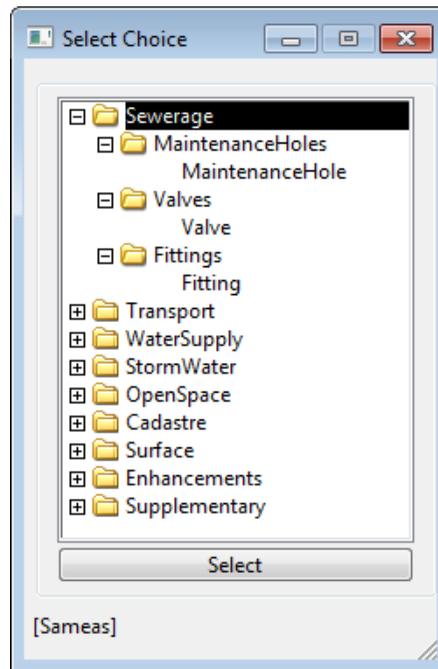
After creating the appropriate ADAC Asset attributes, the ADAC Asset Editor for the particular ADAC Asset is then brought up so any of the created ADAC Asset attributes can be edited.

Selecting **Create asset** brings up the **Create ADAC Asset** panel:

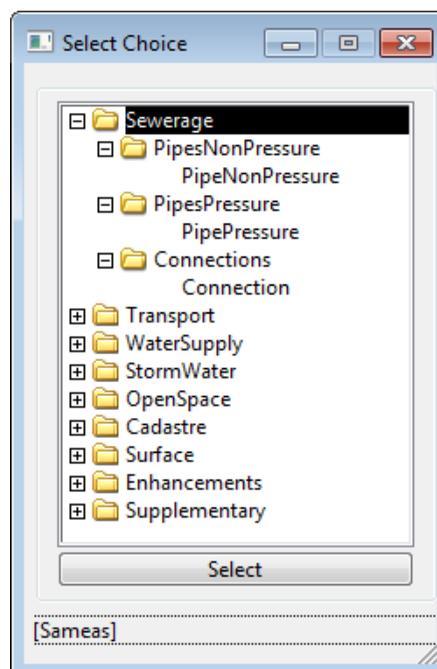


The fields and buttons used in this panel have the following functions.

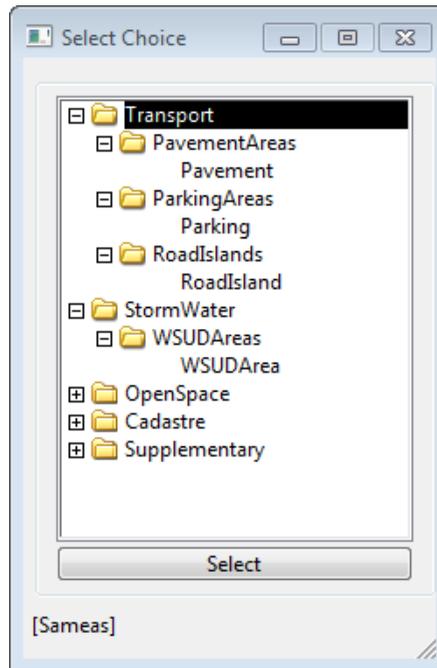
Field Description	Type	Defaults	Pop-Up
<b>String</b> <i>select a string that is to be set up with the ADAC Asset.</i>	string select		
<b>ADAC version</b> <i>the version of ADAC to use when creating the ADAC Header data. The default value is set by the environment variable ADAC_VERSION_4D which is set in the env.4d editor at <b>External Apps &gt;ADAC</b>.</i>	choice box	from env.4d	4.0.0, 4.1.0
<b>ADAC Asset</b> <i>If the geometry of the string selected is a: <b>Point</b> (a one vertex string), then the pop-up only shows those ADAC Assets that have a Point Geometry. For example, for ADAC 4.1, the only ADAC Assets in Sewerage with a Point Geometry are Sewerage&gt;MaintenanceHoles&gt;MaintenanceHole, Sewerage&gt;Valves&gt;Valve and Sewerage&gt;Fittings&gt;Fitting</i>	choice box	depends on the string geometry	



**Polyline** then the pop-up only shows those ADAC Assets that have Polyline Geometry.  
 For example, for ADAC 4.1, the only ADAC Assets in Sewerage with a Polyline Geometry are Sewerage>PipesNonPressure>PipeNonPressure, Sewerage>PipesPressure>PipePressure and Sewerage>Connections>Connection



**Polygon** then the pop-up only shows those ADAC Assets that have Polygon Geometry.  
 For example, for ADAC 4.1, there are NO ADAC Assets in Sewerage with a Polygon Geometry. In Transport there are Transport>PavementAreas>Pavement, Transport>ParkingAreas>Parking and Transport>RoadIslands>RoadIsland



**Asset template** ADAC Asset box available ADAC Asset templates

*the name of an ADAC Asset template that if selected, is used to fill in some of the ADAC Asset attributes for the string.*

*The pop-up looks at the **ADAC Asset** and only displays the templates for that type of ADAC Asset and for the ADAC version.*

**Create** button

*clicking **Create** creates an ADAC Asset structure as 12d ADAC attributes and if there is an **Asset template**, it loads the string's attributes with values from the Asset Template.*

*The [27.4.3 Edit ADAC Header/Asset](#) panel is then brought up so that the ADAC Header data can be create/edited.*

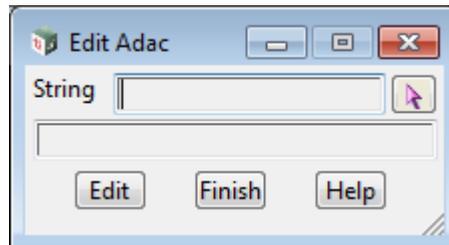
Continue to the next section [27.4.3 Edit ADAC Header/Asset](#) or return to [27.4 12d ADAC Menu](#) or [27 ADAC](#).

## 27.4.3 Edit ADAC Header/Asset

**Position of option on menu:** File I/O =>ADAC =>Edit header/asset

The **ADAC Editor** edits a string and looks at its ADAC attributes and determines whether it is an ADAC **Header** string or an ADAC **Asset** string (and then which **Asset type**), or is not an ADAC string at all.

Selecting **Edit** brings up the **Edit ADAC** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>String</b>	string select		
---------------	---------------	--	--

*select a string to have its ADAC attributes edited.*

<b>Edit</b>	button		
-------------	--------	--	--

*if the selected string is a ADAC Header string, then the [27.4.2 Create ADAC Asset](#) is brought up.*

*If the selected string is an ADAC Asset string, the [27.4.3 Edit ADAC Header/Asset](#) panel is brought up for its ADAC Asset type.*

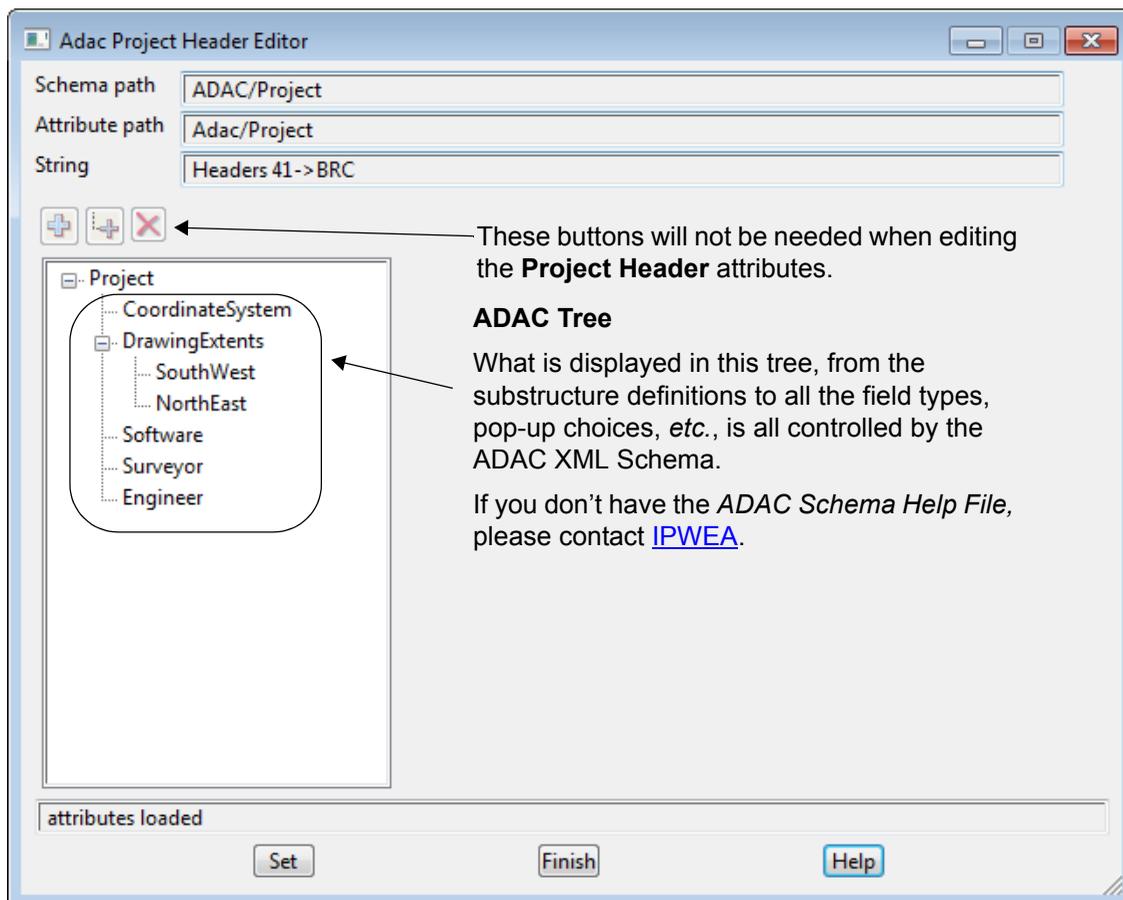
*If the selected string is not an ADAC Header or an ADAC Asset string, then an error message is written to the panel message area.*

Continue to the next section [27.4.3.1 ADAC Header Editor](#) or return to [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

## 27.4.3.1 ADAC Header Editor

**Position of option on menu:** File I/O =>ADAC =>Edit

When a string is selected with the *ADAC Editor* and the string has **ADAC Header Data** attributes, then the **ADAC Project Header Editor** panel is brought up with any ADAC Header Data already with the string loaded into the Editor.



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Schema path</b>	output only		
<i>the path to the ADAC Schema that is used to generate all the items under <b>Project</b>.</i>			
<b>Attribute path</b>	output only		
<i>the path in the ADAC Schema to the ADAC element being edited. In this case it is the ADAC Project element.</i>			
<b>String</b>	output only		
<i>displays the string that has the ADAC Header Data attributes that are being displayed and edited. This will already be set when the panel is brought up by the ADAC Editor or the ADAC Create Header panel.</i>			

### ADAC Tree

*What is displayed in this tree, from the substructure definitions to all the field types, pop-up choices, etc., is all controlled by the ADAC XML Schema.*

*Even what is shown in these images may vary with the ADAC Version.*

*If you don't have the ADAC Schema Help File for the ADAC Version you are creating, please contact [IPWEA](#).*

The **Drawing extents** and **Software** do not have to be filled out because they are updated by **12d Model**. For information about all the other ADAC elements in the **ADAC Tree**, please see the ADAC XML Schema Help file.

**Drawing extents**                      these do not have to be filled in

*these fields do not have to be filled in. The drawing extents will be calculated and updated when the data is written out to an ADAC XML file.*

**Software**                                these do not have to be filled in

*these fields do not have to be filled in. The **Software product** and **Version** will be updated when the data is written out to an ADAC XML file.*

## Buttons at Bottom

**Set**                                        button

*clicking **Set** updates the ADAC Header attributes for the string being edited, using the values in this panel.*

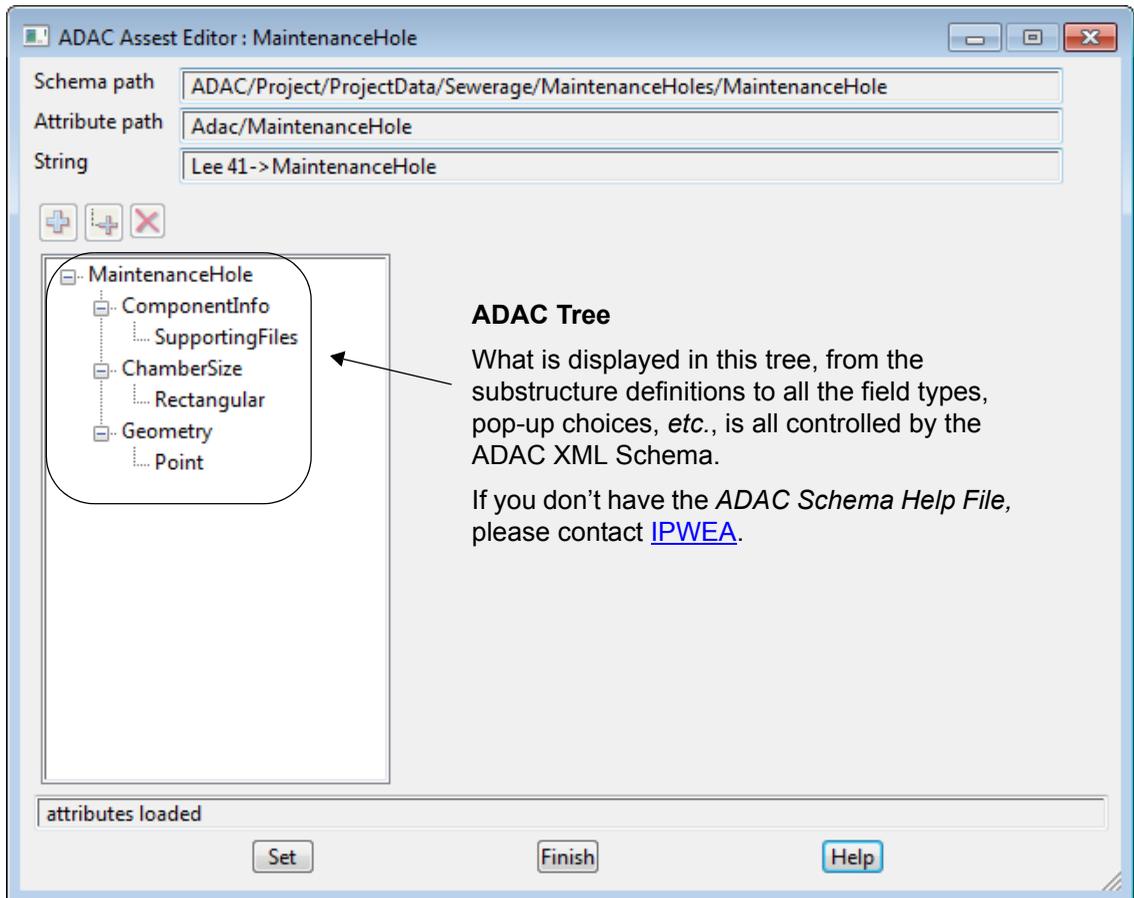
Continue to the next section [27.4.3.2 ADAC Asset Editor](#) or return to [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

## 27.4.3.2 ADAC Asset Editor

**Position of option on menu:** File I/O =>ADAC =>Edit

When a string is selected with the *ADAC Editor* and the string has **ADAC Asset Data** attributes, then the **ADAC Asset Editor** for that particular ADAC Asset is brought up, with any ADAC Asset Data already on the string, loaded into the Editor.

For example, for the ADAC Asset Sewerage>MaintenanceHoles>MaintenanceHole, the **Editor** panel is



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Schema path</b>	output only		
<i>the path to the ADAC Schema that is used to generate all the items for this ADAC Asset.</i>			
<b>Attribute path</b>	output only		
<i>the path in the ADAC Schema to the ADAC element being edited. In this case it is the ADAC Sewerage MaintenanceHole element.</i>			
<b>String</b>	output only		
<i>displays the string that has the ADAC Asset Data attributes that are being displayed and edited. This will already be set when the panel is brought up by the ADAC Editor or the ADAC Create Asset panel.</i>			

### ADAC Tree

*What is displayed in this tree, from the substructure definitions to all the field types, pop up choices, etc., is all controlled by the ADAC XML Schema.*

*Even what is shown in these images may vary with the ADAC Version.*

*If you don't have the ADAC Schema Help File for the ADAC Version you are creating, please contact [IPWEA](#).*

*For information about all the ADAC elements in the **ADAC Tree** for an **ADAC Asset**, please see the **ADAC XML Schema Help file**.*

**Geometry** this do not have to be filled in  
*the Geometry is generated by 12d Model from the strings geometry.*

## **Buttons at Bottom**

**Set** button  
*clicking **Set** updates the ADAC Asset attributes for the string being edited, with the values in this panel.*

Continue to the next section [27.4.3.3 Common Features of ADAC Editors](#) or return to [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

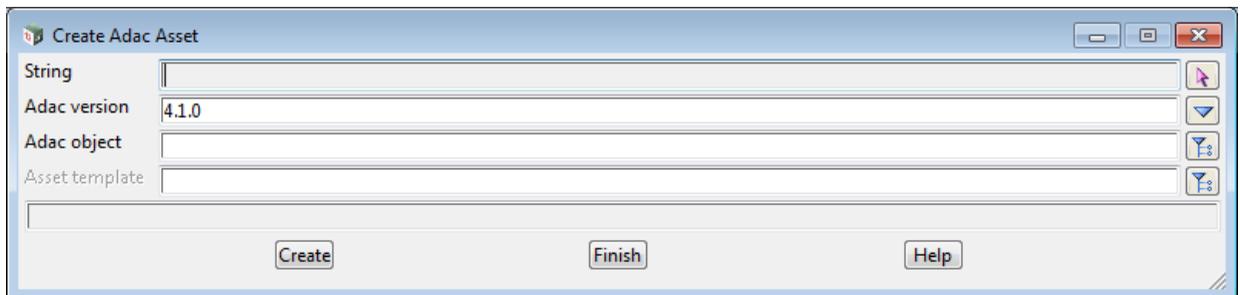
### 27.4.3.3 Common Features of ADAC Editors

The *ADAC Schema* is fully described by the *ADAC XSD (XML Schema Definition)*, which formally defines the structure and all the elements inside an *ADAC XML* file.

And the **ADAC Header Editor** and the **ADAC Asset Editors** for each of the different *ADAC Assets*, are also fully defined by the *ADAC Schema XSD*.

What that means is that everything in the *ADAC Editors*, and how they behave, follows the *ADAC XSD*.

That starts with **Create ADAC Asset** panel:



where once a string is selected, the choice of **ADAC Asset** choices is totally determined by which *ADAC Assets* in the *ADAC XSD* have that geometry.

Once in the **Editors** (*ADAC Asset* or *ADAC Header*), every item in the *Editor* is again controlled by the *ADAC XSD*.

So to know why each items exists, and what it is, you need to refer to the *ADAC Schema*.

What we'll now do is look at how the Editor is structured, navigated and used.

See

[27.4.3.3.1 Editor Tree, Nodes and Elements](#)

[27.4.3.3.2 Delete Icon](#)

[27.4.3.3.3 Add Sibling and Add Child Icons](#)

[27.4.3.3.4 Nillable Elements and Nodes](#)

[27.4.3.3.5 Nodes with Choices for Subnodes](#)

### 27.4.3.3.1 Editor Tree, Nodes and Elements

When the ADAC Editors start up, they have a tree structure on the left hand side.

Each item in the tree are known as nodes and **nodes** contain ADAC elements or can have their own substructure of nodes and elements. A node with a substructure is identified by having either a **+** or a **-** to the left of it.

The **+** indicates that the node has a substructure that has not been expanded and by clicking on the **+**, the first level of the sub structure is displayed.

The **-** indicates that the node contains a substructure and it has been expanded in the tree. Clicking on the **-** will collapse the expansion back into just the node.

When a node is clicked on and hence highlighted, all the element that belong to the first level of the node that are not other nodes, are displayed on the right hand side of the panel.

It is the non-node elements displayed on the right that hold the data about the ADAC Asset.

The - indicates that it is a node. Clicking on the - will collapse everything back to the node.

The + indicates that it is a node. Clicking on the + will expand the first level of the node.

The non-node elements for the highlighted node are displayed on the right hand side of the panel.

**ADAC Tree**

What is displayed in this tree, from the substructure definitions to all the field types, pop-up choices, etc., is all controlled by the ADAC XML Schema.

If you don't have the ADAC Schema Help File, please contact [IPWEA](#).

ObjectID	Text icon	Choice icon
Use	Overflow	
SurfaceLevel_m	null	
InvertLevel_m	null	
FloorConstruction	Prefabricated	
FloorMaterial	Concrete	
WallConstruction	Prefabricated	
WallMaterial	Concrete	

**Choice icon**

is valid

Set Finish Help

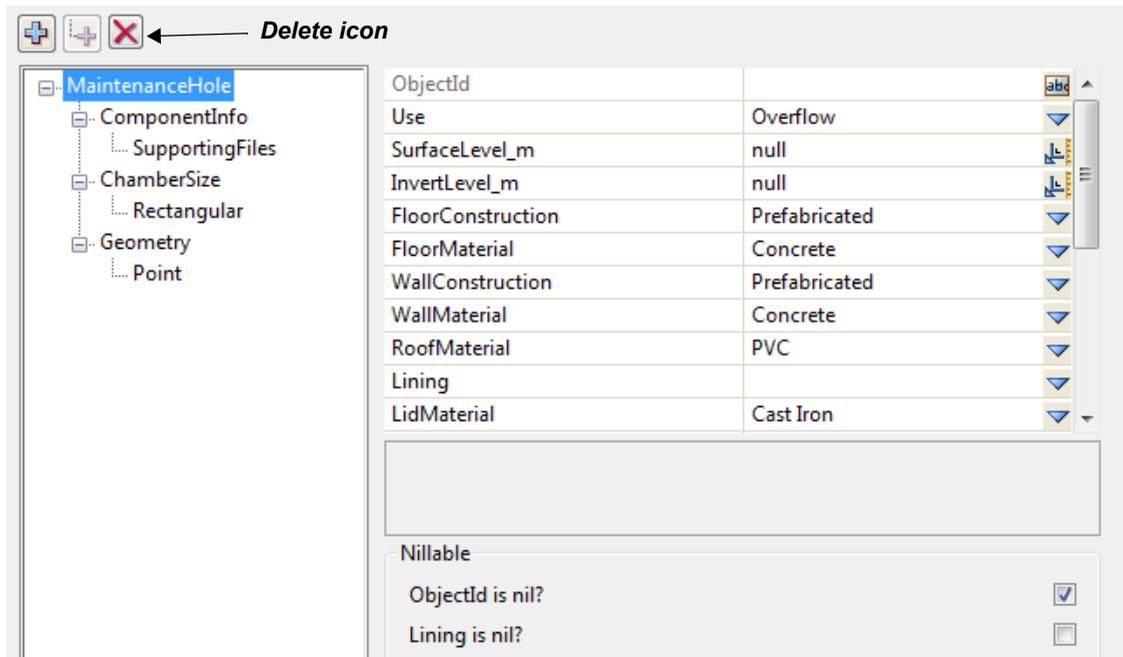
The elements on the right hand side of the panel mainly consist of choice, text, real, integer and date boxes and they are all defined in the **ADAC XSD**. Where ever possible, the icons at the end of each field are the same as those used in **12d Model** panels.

Where it is a **Choice** box, the allowable choices (which are in the pop up) are all defined in the **ADAC XSD** where they are known as **Enumerations**.

Continue to the next section [27.4.3.3.2 Delete Icon](#), or return to [27.4.3.3 Common Features of ADAC Editors](#), [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

### 27.4.3.3.2 Delete Icon

There is a **Delete** icon on the ADAC Editors but most of the time is disabled because there are only a few places in the ADAC Schema that data can be deleted.



If a node in the tree is selected and highlighted, and the **Delete** icon is enabled and then clicked, the node is deleted.

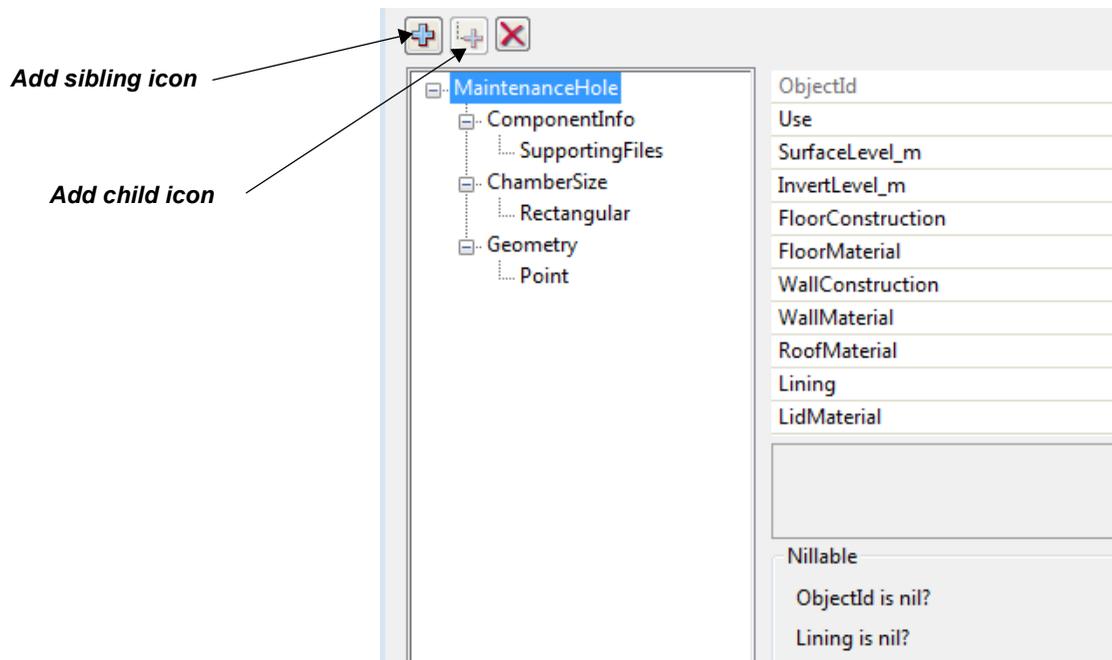
The **Delete** icon is enabled when the ADAC Asset itself has been selected as in the image above but deleting the ADAC Asset itself is rarely done.

The one time when the **Delete** is usable is when a node has choices for one of its subnodes (for example, ChamberSize). In that case to change the choice of subnode, the existing subnode needs to be deleted and then a new one created. replaced by a new one. This process is documented in the section [27.4.3.3.5 Nodes with Choices for Subnodes](#).

Continue to the next section [27.4.3.3.3 Add Sibling and Add Child Icons](#) or return to [27.4.3.3 Common Features of ADAC Editors](#), [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

### 27.4.3.3.3 Add Sibling and Add Child Icons

There are two Add icons - **Add** a sibling and **Add** a child.



The **Add a sibling** icon adds a new item at the **same level** (a sibling) as the highlighted node.

The **Add a child** icon adds a node as a **subnode** of the highlighted node.

As in the **Delete** icon case, most of the time the **Add a sibling** and **Add a child** icons are disabled because there are only a few places in the *ADAC Schema* that it makes sense to add a new sibling or child node.

The **Add a sibling** icon is enabled when the *ADAC Asset* itself has been selected as in the image above but adding a second *ADAC Asset* is not useful because a second or subsequent *ADAC Asset* on the one string will be ignored in the 12d-ADAC process.

One time when the **Add a child** is usable is when a node has choices for one of its subnodes (for example, *ChamberSize*). In that case to change the choice of subnode, the existing subnode needs to be deleted and then a new one created. replaced by a new one using **Add a child**. This process is documented in the section [27.4.3.3.5 Nodes with Choices for Subnodes](#).

Continue to the next section [27.4.3.3.4 Nillable Elements and Nodes](#) or return to [27.4.3.3 Common Features of ADAC Editors](#), [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

#### 27.4.3.3.4 Nillable Elements and Nodes

One concept used in the ADAC XSD that has not been expressed in the same way in other **12d Model** panels, is **nillable**.

In the ADAC XSD, if a *node* or *element* is **nillable** (i.e. has **nil** set to true in the XSD), then the node or element may or may not have a value. So it is optional.

If a *node* or *element* is **not nillable** (i.e. has **nil** set to **false** in the XSD) then the node or element is not optional and must have values.

See

[27.4.3.3.4.1 Nillable Elements](#)

[27.4.3.3.4.2 Nillable Node](#)

Or return to [27.4.3.3 Common Features of ADAC Editors](#), [27.4.3 Edit ADAC Header/Asset](#), [27.4 12d ADAC Menu](#) or [27 ADAC](#).

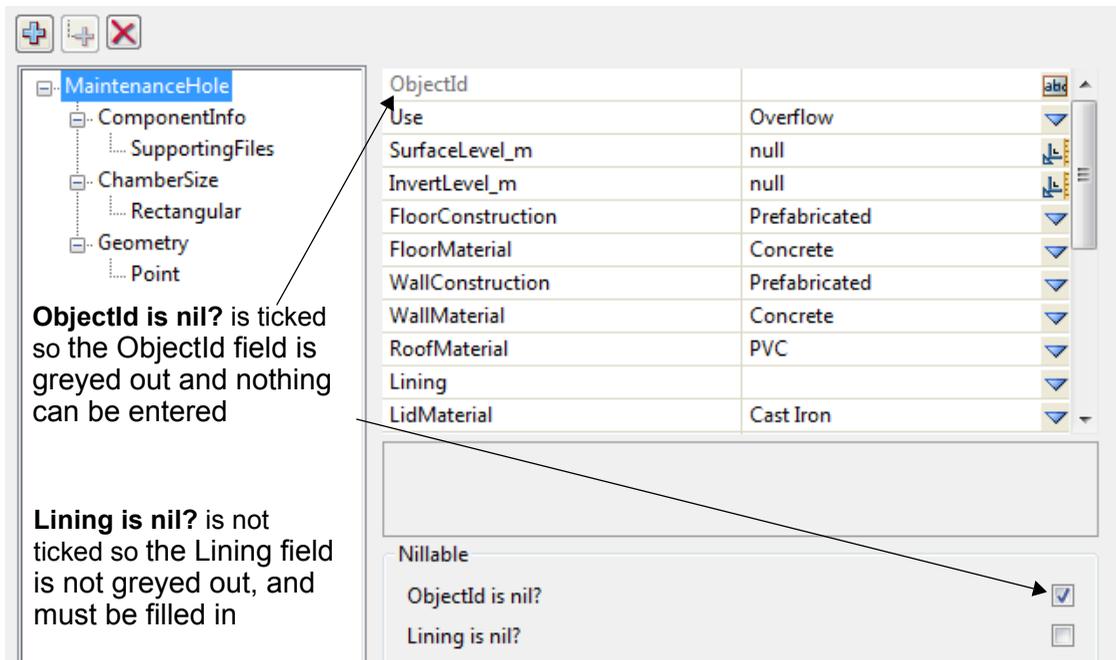
### 27.4.3.3.4.1 Nillable Elements

In the ADAC **Editor**, if an element is **nillable** (and hence optional) then it is listed on the right hand side of the panel under the title **Nillable** and with a tick box beside its name.

If the tick box is **ticked** then the item is not to be filled in and the panel field for it is greyed out so nothing can be entered.

If the tick box is **not ticked** then the item is to be filled in and the panel field is no longer greyed out.

All elements not mentioned in the Nillable area are **not nillable** and must always be filled in.



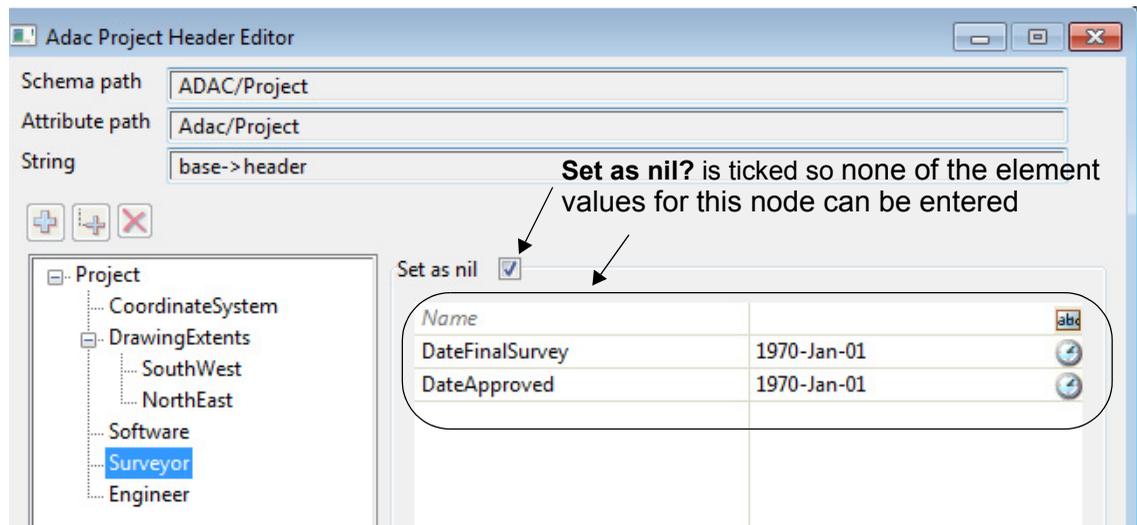
Continue to the next section [27.4.3.3.4.2 Nillable Node](#) or return to [27.4.3.3.4 Nillable Elements and Nodes](#).

### 27.4.3.3.4 Nillable Node

In the ADAC **Editor**, if a node **is nillable** (and hence optional) then on top of the listing of the elements for the node is a tick box **Set as nil**.

If the tick box is **ticked** then none of the elements can be filled in.

If the tick box is **not ticked** then the element have the potential of being filled it. Whether they are to be filled in depends on if the individual element is nillable or not.



Return to [27.4.3.3.4 Nillable Elements and Nodes](#).

### 27.4.3.3.5 Nodes with Choices for Subnodes

In the *ADAC Schema*, there are some nodes that have choices for subnodes.

For example, **Sewerage>MaintenanceHoles>MaintenanceHole>ChamberSize** is not just a since value but is a choice of subnodes - *Rectangular*, *Circular* and *Custom* - which each contain one or more elements.

**Element: ChamberSize**  
 Occurrences: Minimum = 1 (Required) Maximum = 1  
 Can be Nil = false

*Data structure describing the chamber configuration and dimensions.*

**Internal Complex Data Type Constraining: [ChamberSize]**

*No documentation provided*

Choice Occurrences: Minimum = 1 (Required) Maximum = 1

**Choice**

**Element: Rectangular** { Data Type: [rectangular\\_struct\\_mm](#) }  
 Occurrences: Minimum = 1 (Required) Maximum = 1  
 Can be Nil = false

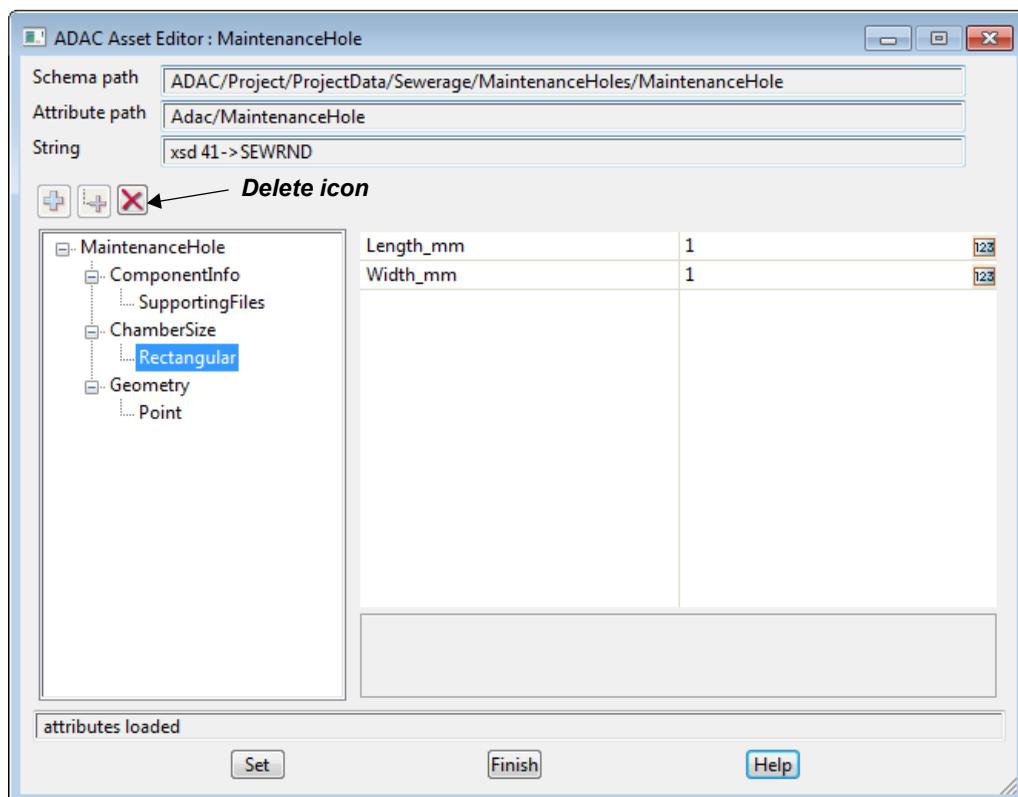
*Data container for rectangular dimensions*

**Element: Circular** { Data Type: [circular\\_struct\\_mm](#) }  
 Occurrences: Minimum = 1 (Required) Maximum = 1  
 Can be Nil = false

*Data container for circular dimensions*

**Element: Custom** { Data Type: [custom\\_area\\_struct\\_sqm](#) }  
 Occurrences: Minimum = 1 (Required) Maximum = 1  
 Can be Nil = false

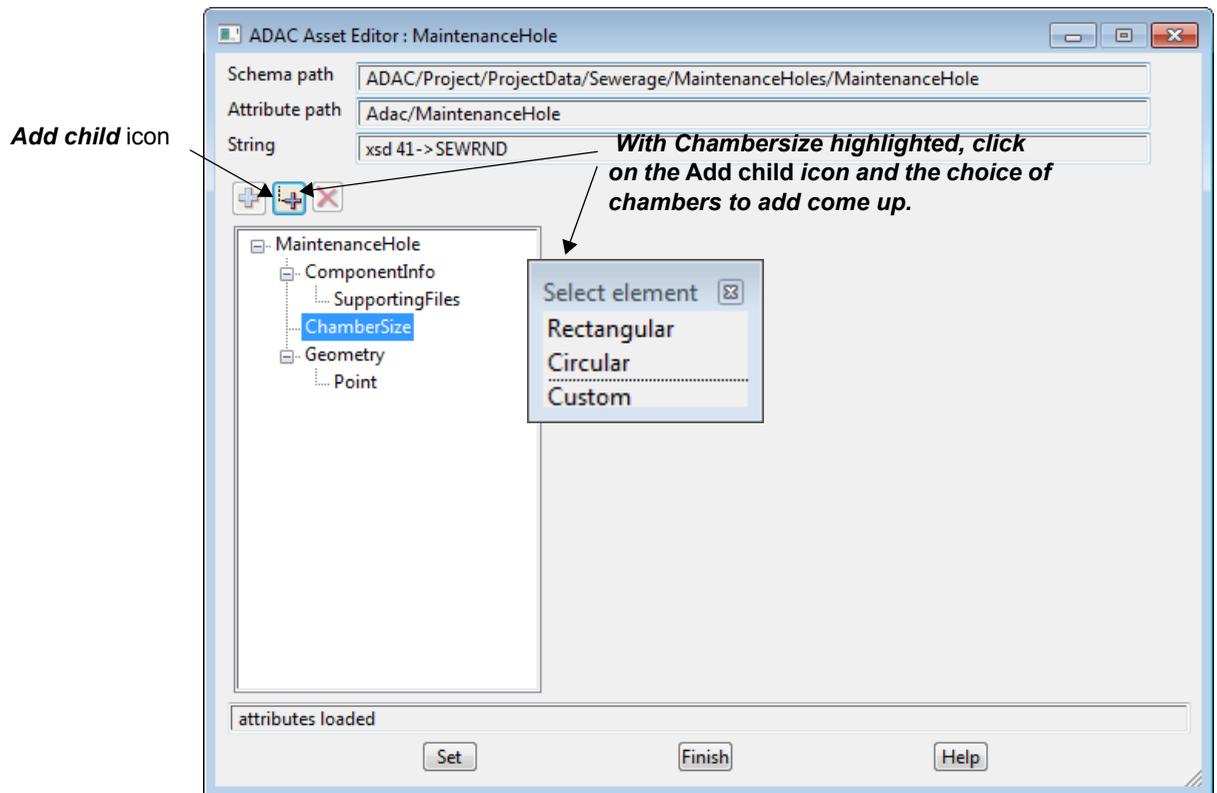
*Custom Shaped chamber. Such a feature should be associated with a plan or document describing its layout and dimensions*



To change the subnode choice, you need to click on the **subnode** to highlight it and then click on the **Delete** icon.

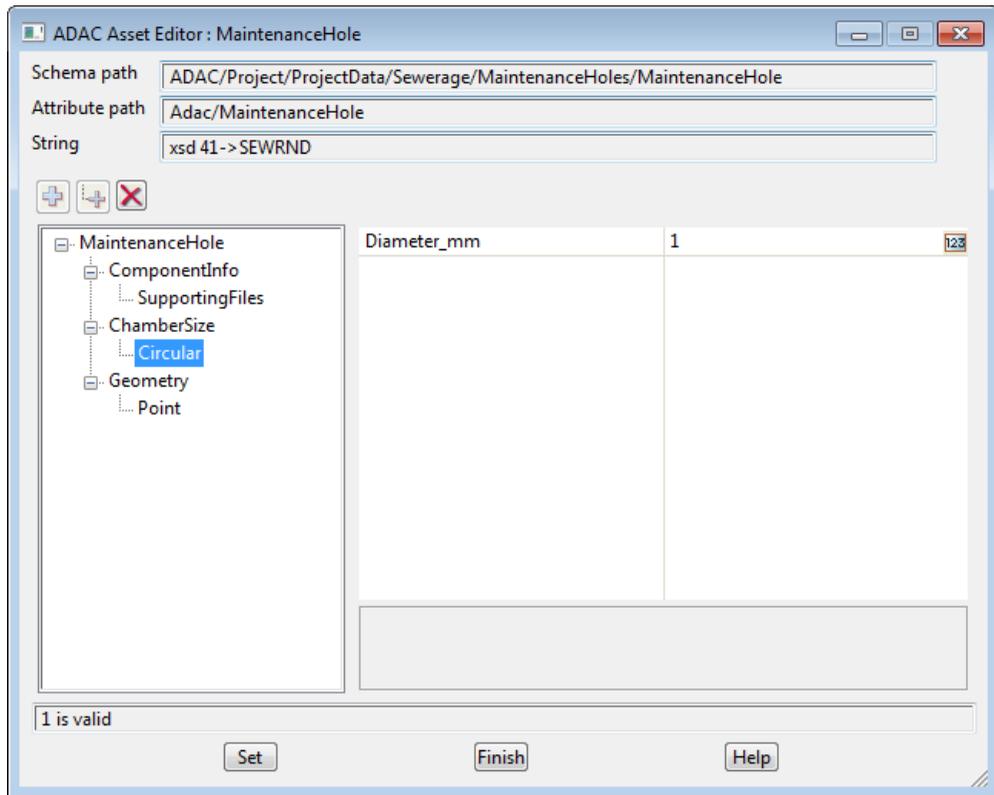
To create a new subnode now that one no longer exists, click on the **node** to highlight it and then click on the **Add Child** icon. A **Select Element** panel will then appear with the available choices.

For example, for the **Sewerage>MaintenanceHoles>MaintenanceHole>ChamberSize**, the choices are *Rectangular*, *Circular* and *Custom*.



Select the choice from the **Select Element** panel and the new **subnode** is created for the highlighted node.

For example, if *Circular* is chosen as the choice of subnode of *ChamberSize*



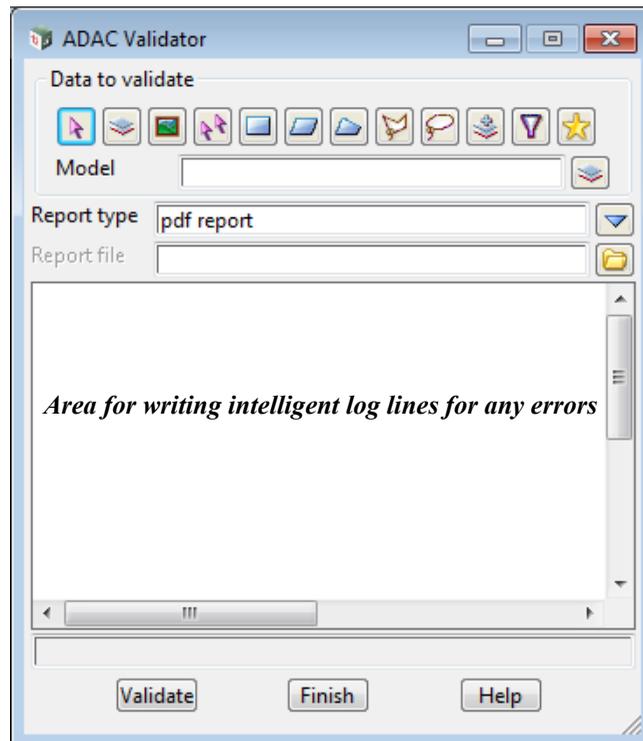
Return to [27.4.3.3 Common Features of ADAC Editors](#).

## 27.4.4 Validate

Position of option on menu: **File I/O =>ADAC =>Validate**

The **ADAC Validator** validates a *Data Source* of string against the ADAC XML Schema.

Selecting **Validate** brings up the **ADAC Validator** panel:

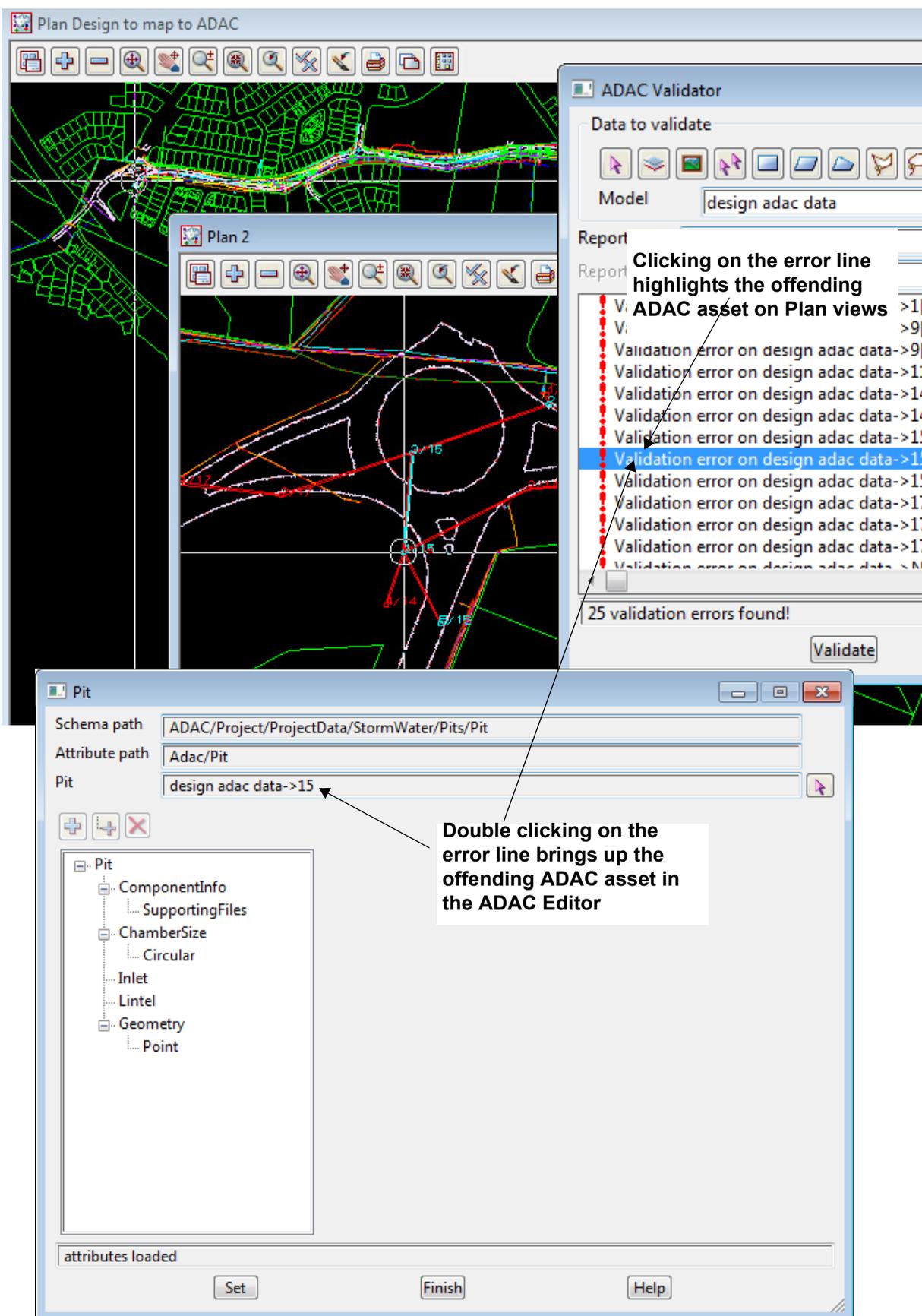


The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of data to be validated against the ADAC Schema.</i>			
<b>Report type</b>	choice box	ADAC pdf, ADAC html, original xml, custom	
<i>output format for the report. An XML file will be produced and then if required, converted to the selected report.</i>			
<i>For information on setting up custom reports from the generated XML file using xslts, see <a href="#">4.30 Setting Up XML Reports</a>.</i>			
<b>Report file</b>	file box		
<i>if <b>not blank</b>, an XML file will be created and a report of this name, and of the type given by <b>Report type</b> will be generated from the XML file.</i>			
<i>If <b>blank</b>, no report is created but errors are still written to the panel's message area.</i>			
<b>Validate</b>	button		
<i>when <b>Validate</b> is pressed, the string in the Data Source is validated against the ADAC XML Schema.</i>			
<i>If no errors are found, then <b>No errors found!</b> is written to the panel's message area.</i>			
<i>For each error found, an error message is written to the area on the panel for error messages as an intelligent log line, and if <b>Report file</b> is not blank, written in XML to the report file.</i>			

*Clicking on an error line will highlight and pan to the offending ADAC Asset in any Plan view the asset is on.*

*Double clicking on the log line, brings up the ADAC Asset Editor with the data for the string loaded into it.*



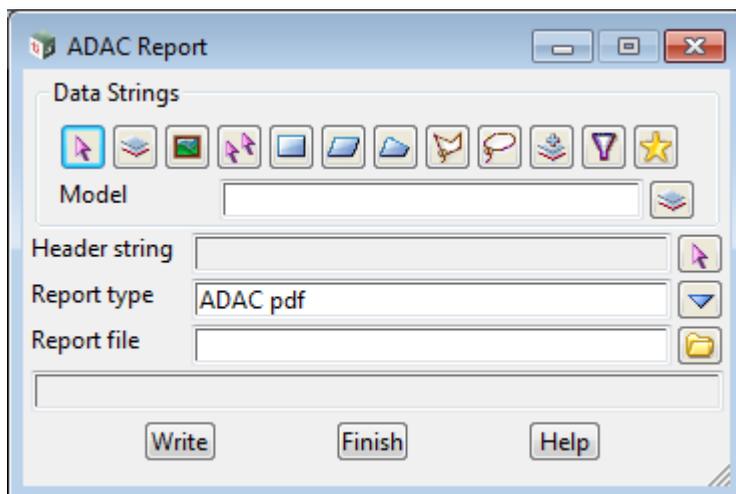
Continue to the next section [27.4.5 Report](#) or return to [27.4 12d ADAC Menu](#).

## 27.4.5 Report

Position of option on menu: **File I/O =>ADAC =>Report**

**Report** produces a report in a variety of formats of a ADAC Header string and a user selection of ADAC Asset strings.

Selecting **Report** brings up the **ADAC Report** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to be included in the ADAC report.*

**Header string**

string select

*select the string that has the ADAC Header information for this report.*

**Report type**

choice box

ADAC pdf, ADAC html, original xml, custom

*output format for the report. An XML file will be produced and then if required, converted to the selected report.*

*For information on setting up custom reports from the generated XML file using xslts, see [4.30 Setting Up XML Reports](#).*

**Report file**

file box

*if **not blank**, an XML file will be created and a report of this name, and of the type given by **Report type** will be generated from the XML file.*

*If **blank**, no report is created.*

**Write**

button

*writes out the report.*



## 12D ADAC REPORT

Current Date: 31-01-2014, 19:00:01

<b>Project name:</b>	X	<b>Work Approval ID:</b>	
<b>ADAC version:</b>	4.0.0	<b>Drawing Number:</b>	X
<b>Description:</b>	x	<b>Drawing Revision:</b>	
<b>Status:</b>	Preliminary	<b>Construction date:</b>	
<b>Export date time:</b>	2013-06-05T15:46:35	<b>Software product:</b>	12d Model 11.0 Alpha 301 Lees tinkers 16 - Not For Production
<b>Owner:</b>		<b>Software version:</b>	11.0.0.301
<b>Receiver:</b>	X		

	Surveyor	Engineer
<b>Name:</b>		
<b>Date approved:</b>		
<b>Date final survey:</b>		

### Coordinate system

**Horizontal coordinate:** x  
**Horizontal datum:** x  
**Vertical datum:** x  
**Is approximate:** false  
**Origin mark:**  
**Notes:**

### Drawing Extents

			GNSS Metadata						
	X	Y	X	Y	Z	Time	H_prec	V_prec	Std dev
<b>South West</b>	68920.586025	79890.501585							
<b>North East</b>	73334.968672	82603.065224							

### Transport

#### Road Edge:

<b>Object ID</b>	14797225	14797227	14797229	14797232
<b>Infrastructure code</b>				
<b>Owner</b>				
<b>Status</b>	As-Constructed	As-Constructed	As-Constructed	As-Constructed
<b>Notes</b>				
<b>Supporting File</b>				
<b>Type</b>	Bitumen	Bitumen	Bitumen	Bitumen
<b>Length(m)</b>	87.788	127.719	72.756	25.983
<b>Pavement Extension(mm)</b>	1	1	1	1

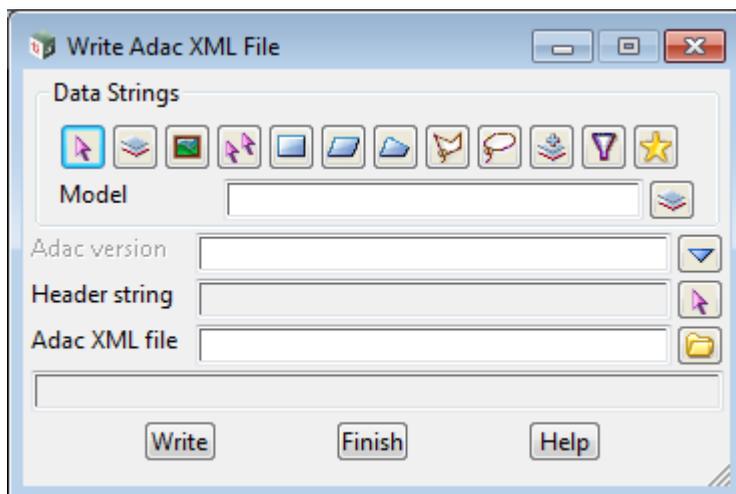
Continue to the next section [27.4.6 Write ADAC XML File](#) or return to [27.4 12d ADAC Menu](#) .

## 27.4.6 Write ADAC XML File

Position of option on menu: **File I/O =>ADAC =>Write ADAC file**

**Write ADAC file** produces an ADAC XML file using a user selected ADAC Header string and a user selection of ADAC Asset strings.

Selecting **Write ADAC file** brings up the **Write ADAC XML file** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of ADAC Asset data to write out to an ADAC XML file.*

**Header string**

string select

*select the string that has the ADAC Header information for this ADAC.XML file*

**Report file**

file box

*name of the file to write the ADAC XML data to.*

**Write**

button

*write out the ADAC XML file using the Header string as the ADAC Header data and the Data source of ADAC Asset strings.*

Continue to the next section [27.4.7 Import ADAC XML File](#) or return to [27.4 12d ADAC Menu](#).

## 27.4.7 Import ADAC XML File

Position of option on menu: **File I/O =>ADAC =>Import ADAC file**

**Import ADAC file** reads in an ADAC XML file and

- (a) For the **ADAC Header Information**, creates a **12d Model** one vertex string with an identical attribute structure to an ADAC Header string created in **12d Model**. All the ADAC Header data in the ADAC XML file is placed in the attribute structure.
  - (i) The string is given the **string name header**.
  - (ii) This string is placed in the model **adac project header** with a **Pre\*postfix for models** from the panel applied to the model name.
- (b) For each **ADAC Asset** in the file, creates a **12d Model** string with the same geometry as the ADAC Asset has in the ADAC XML file.
  - (i) The **12d Model** string is given an identical attribute structure to an ADAC Asset created within **12d Model**. All the ADAC Asset data in the ADAC XML file for that asset is read in and placed in the attribute structure.
  - (ii) The created string is given the **final** part of the ADAC Asset name as a **string name**. For example, A *Transport>RoadEdges>RoadEdge* is given the name *RoadEdge*.
  - (iii) The string is placed in the model **adac project data** with a **Pre\*postfix for models** from the panel applied to the model name.

It is unfortunate that with only sixty-eight ADAC Assets, there is a double up in such names because it is much easier to refer to *MaintenanceHole* than *Sewerage>MaintenanceHoles>MaintenanceHole*.

In ADAC 4.1, the double ups are:

**MaintenanceHole** - in both *Sewerage* and *WaterSupply*

**Valve** - in both *Sewerage* and *WaterSupply*

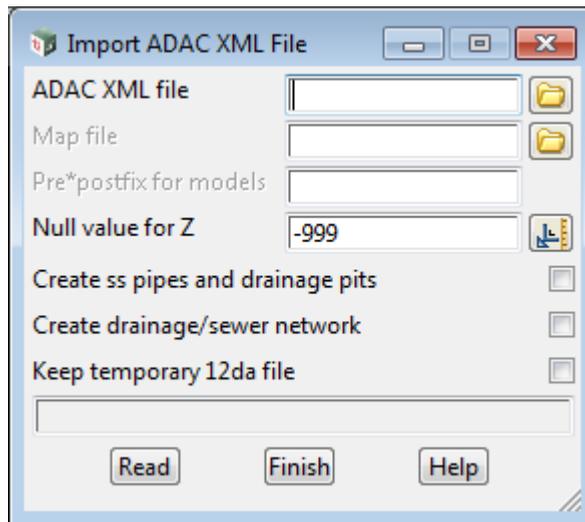
**Fitting** - in all three of *Sewerage*, *WaterSupply* and *StormWater*

**Pipe** - in both *WaterSupply* and *StormWater*

**Connection** - in both *Sewerage* and *Cadastre*.

To allow them to be distinguished, when **12d Model** reads the ADAC Asset data, the **full name of the ADAC Asset** is stored in the string text attribute **Adac>Path**. So if a *Map File* is used when **reading in** the ADAC.MXL file, using this attribute as the **Att Key** in the **Basic** node allows the string's ADAC Asset type to be uniquely identified, and the string **renamed**.

Selecting **Import ADAC file** brings up the **Import ADAC XML file** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>ADAC XML file to import</b>	file box		available .XML files

*name of the ADAC XML file to import.*

*The default model names (with the **Pre\*postfix for models** applied) to hold the data read in are:*

*adac project header for the string containing the ADAC Header data.*

*adac project data for the strings for each ADAC Asset*

<b>Map file</b>	file box		*.mapfile, *.mf files
-----------------	----------	--	-----------------------

*if **blank**, no 12d Map File is used*

*If **not blank**, the name of the **Map File** to be used when importing data from the ADAC.XML file.*

*When using a **Map File**, the short **ADAC Asset** name in the ADAC XML file is used as the entity-name for matching with the key in the Map File. For example, a **Transport>RoadEdges>RoadEdge** is given the name **RoadEdge**.*

*That is, when the ADAC Asset is read in, it can be mapped by having an entry in the **Base** node of the Map File with short ADAC Asset name as the **Key**.*

*For example, **RoadEdge**, or **RoadEdge\*** as the **Key** will map all the **Transport>RoadEdges>RoadEdge** that are read in.*

*When importing, all the ADAC elements associated with the ADAC Asset are placed in a 12d attribute structure identical to that used when ADAC Assets are created within **12d Model**. So the ADAC Asset attributes are also available to use in the **Base** node of the Map File as an **Att Key** when importing the ADAC XML file.*

*Note that the **full** ADAC Asset name is stored in the string attribute **Adac>Path** so this is available to use as the **Att Key** to distinguish between ADAC Assets with the same short name.*

*See the section [9.8.1 Create/Edit a Map File](#) for information about 12d map files.*

<b>Pre*postfix for models</b>	pre*postfix box	name of xml file (without the XML)	
-------------------------------	-----------------	------------------------------------	--

*When an XML is selected, the file name (without the .xml) plus " \* " is written to this field as the default.*

*if **not blank**, a prefix and a postfix applied to the default model names that are created.*

*If there is a 12d Map File then this will also be applied to any models created by the map file.*

*Go to the section [4.19.2 Pre\\*Postfix in Panel Fields](#) for information on using pre\*postfix.*

<b>Null value for Z</b>	real box	-999	
-------------------------	----------	------	--

*if in the ADAC geometry there is no z-coordinate, then the coordinate is given this value in 12d Model.*

**Create ss pipes and drainage pits** tick box

*if **not ticked**, the string created by ADAC are super strings with no diameters.*

*If **ticked**, ADAC pipes are created as super strings with round or box sections.*

*If **ticked and you have the drainage module**, Sewerage MaintenanceHoles and Stormwater Pits create a pit on a one vertex drainage string, and the pit geometry is defined by the ADAC elements.*

**Create drainage network** tick box

*If **ticked and you have the drainage module**, 12d Model will try to join the Sewerage MaintenanceHoles and NonPressurePipes into Sewer lines, and the Stormwater Pits and Pipes into drainage lines.*

*However, ADAC has no information about which Pits/MaintenanceHoles go with which Pipes, so the results can only be a best guess.*

*If the ADAC XML file was created by 12d Model from 12d Model drainage and sewer strings, then extra information is added to the ADAC XML data so that the original networks can be reconstructed.*

*When drainage strings are created, they are given a name starting with **StormWater** so to map them when reading in the data you will need an entry in the **Base** node of the Map File with **StormWater\*** as the **Key**.*

*When sewer strings are created, they are given a name starting with **Sewerage** so to map them when reading in the data you will need an entry in the **Base** node of the Map File with **Sewerage\*** as the **Key**.*

**Keep temporary 12da file** tick box

*when an ADAC XML file is being imported, a temporary 12da file is created.*

*If **not ticked**, the temporary 12da file is deleted.*

*If **ticked**, the temporary 12da file is not deleted.*

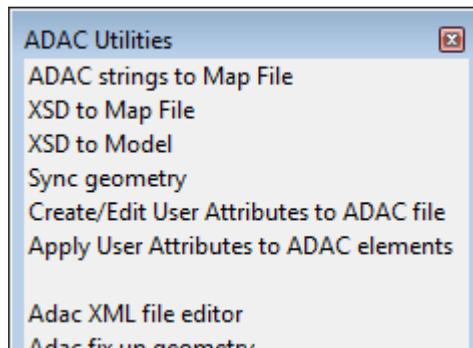
**Import** button

*when clicked, the ADAC XML file is processed and the ADAC data imported into 12d Model.*

Continue to the next section [27.4.8 ADAC Utilities](#) or return to [27.4 12d ADAC Menu](#).

## 27.4.8 ADAC Utilities

**Position of menu:** File I/O =>ADAC =>Utilities



See

[27.4.8.1 ADAC Strings to Map File](#)

[27.4.8.2 XSD to Map File](#)

[27.4.8.3 XSD to Model](#)

[27.4.8.4 Sync Geometry](#)

[27.4.8.5 Create/Edit User Attributes to ADAC File](#)

[27.4.8.6 Apply User Attributes to ADAC Elements](#)

[27.4.8.7 ADAC XML File Editor](#)

## 27.4.8.1 ADAC Strings to Map File

**Position of option on menu:** File I/O =>ADAC =>Utilities =>ADAC strings to Map File

**ADAC strings to Map File** takes the selected strings representing *ADAC Assets* and creates a **Map File** from them in the following way:

For applications that **don't** have **12d Model** drainage or sewer strings:

each ADAC string creates a row in the *Attributes>String* grid of the **Map File** with the **string name** followed by a \* placed in the **Name** column. The first string attribute's name, type and value is placed in the **Att Key** column, and the **ADAC attribute group** placed in the **Map Attributes** column.

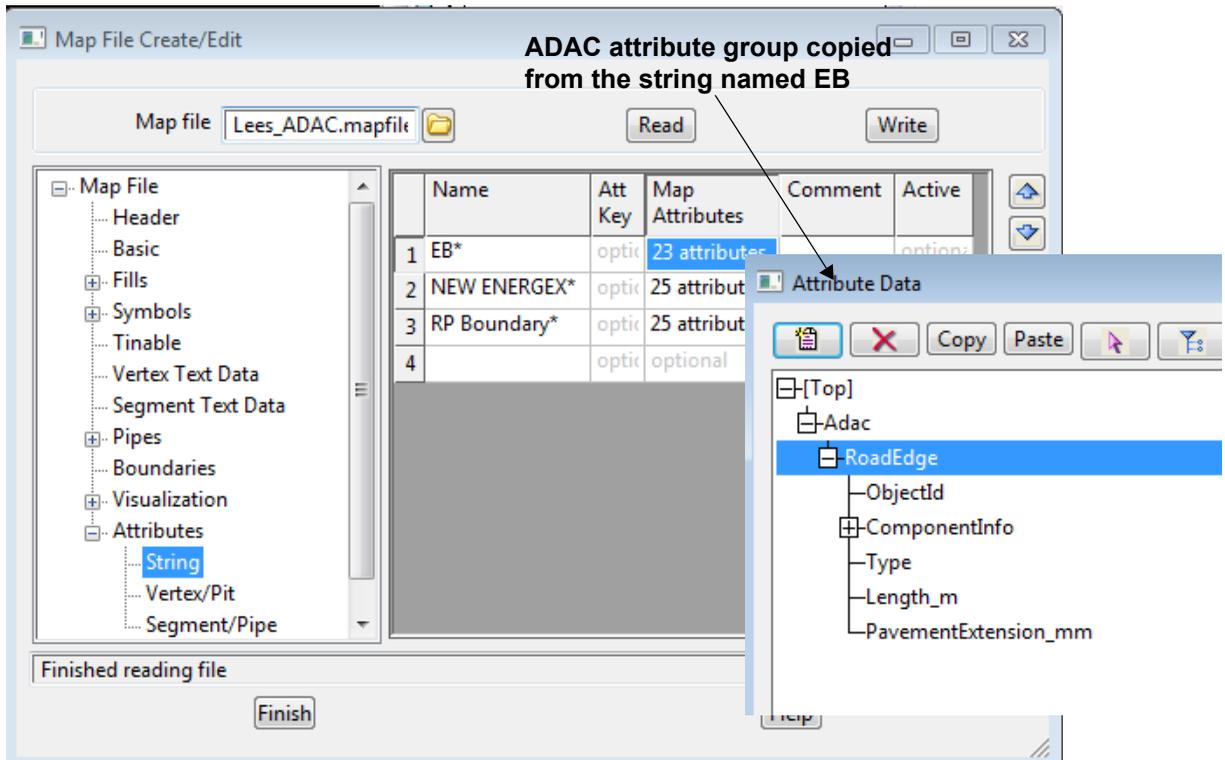
For applications that **do** have **12d Model** drainage or sewer strings then have the **User Vertex and segment attributes** tick box **ticked** and

- (a) each ADAC *Sewerage>MaintenanceHoles>MaintenanceHole* string creates a row in the *Attributes>Vertex/Pit* grid of the **Map File** with \* placed in the **Name** column, an Integer attribute named **sewer type** with value **1**, placed in the **Att Key** column. The first string attribute's name, type and value is placed in the **Vertex Att Key** column and the **ADAC attribute group** placed in the **Map Attributes** column. The string name is written to **Comment** column.
- (b) each ADAC *StormWater>Pits>Pit* string creates a row in the *Attributes>Vertex/Pit* grid of the **Map File** with \* placed in the **Name** column, an Integer attribute named **sewer type** with value **0**, placed in the **Att Key** column. The first string attribute's name, type and value is placed in the **Vertex Att Key** column and the **ADAC attribute group** placed in the **Map Attributes** column. The string name is written to **Comment** column.
- (c) each ADAC *Sewerage>PipesNonPressure>PipeNonPressure* string creates a row in the *Attributes>Segment/Pipe* grid of the **Map File** with \* placed in the **Name** column, an Integer attribute named **sewer type** with value **1**, placed in the **Att Key** column. The first string attribute's name, type and value is placed in the **Segment Att Key** column and the **ADAC attribute group** placed in the **Map Attributes** column. The string name is written to **Comment** column.
- (d) each ADAC *StormWater>Pipes>Pipe* string creates a row in the *Attributes>Segment/Pipe* grid of the **Map File** with \* placed in the **Name** column, an Integer attribute named **sewer type** with value **0**, placed in the **Att Key** column. The first string attribute's name, type and value is placed in the **Segment Att Key** column and the **ADAC attribute group** placed in the **Map Attributes** column. The string name is written to **Comment** column.
- (e) all other ADAC strings create a row in the *Attributes>String* grid of the **Map File** with the **string name** followed by a \* placed in the **Name** column, and the first string attribute's name, type and value placed in the **Att Key** column, and the **ADAC attribute group** placed in the **Map Attributes** column.

For example, if you had a model with three ADAC Assets in it

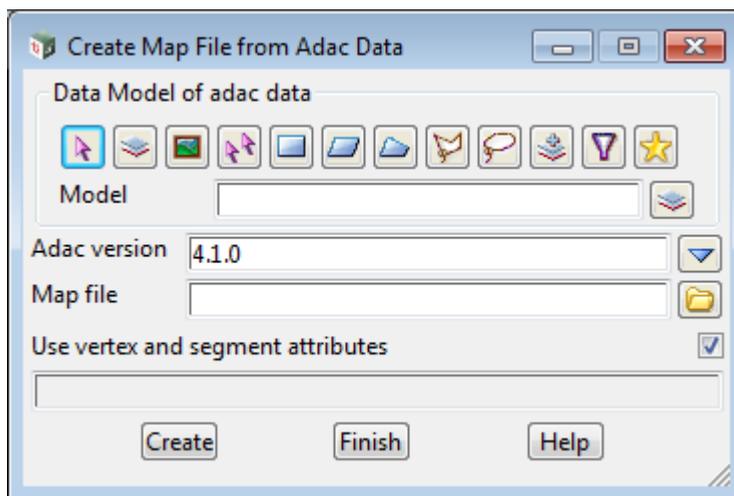
- (a) a Road Edge with the string name EB
  - (b) an ElectricalConduit with string name NEW ENERGEX
- and
- (c) a Lot with string name RP Boundary

then running the option **Assets to Map File** creates a **Map File** with three entries in the *Attributes >Strings* section:



**Assets to Map File** is the best method for creating a **12d Map File** to be used in the ADAC Survey and Designers Chains.

Selecting **Assets to Map file** brings up the **Create Map File from ADAC Data** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of ADAC Asset data to create a 12d Map File from.</i>			

**ADAC version** choice box from env.4d 4.0.0, 4.1.0

*this field is optional, but if it is set, then only strings representing ADAC Assets of this version of the ADAC XML Schema will have entries created in the Map File.*

**Map file** file box \*.mapfile, \*.mf files

*the name of the **12d Map File** that is created and has the **Attributes >String** section.*

*For each string that is an ADAC Asset of the ADAC version given in the **Version** field, create a row in the **Attributes >String** section of the this **Map File** with the name of the string followed by an \* in the **Name** column, and in the same row, all the string's string attributes in the Adac group are copied to the **Map Attributes** column*

*See the section [9.8.1 Create/Edit a Map File](#) for information about 12d map files.*

**User vertex and segment attributes** tick box not ticked

*if **not ticked**, each ADAC string creates an entry in the **Attributes >String** section of the Map File.*

*If **ticked**, ADAC Sewerage>MaintenanceHoles>MaintenanceHole and StormWater>Pits>Pit create entries in the **Attributes>Vertex/Pit** section of the Map File, ADAC*

*Sewerage>PipesNonPressure>PipeNonPressure and StormWater>Pipes>Pipe create entries in the **Attributes>Segment/Pipe** section of the Map File, and all other ADAC strings create entries in the **Attributes >String** section of the Map File.*

**Create** button

*when clicked, the **Map File** is created.*

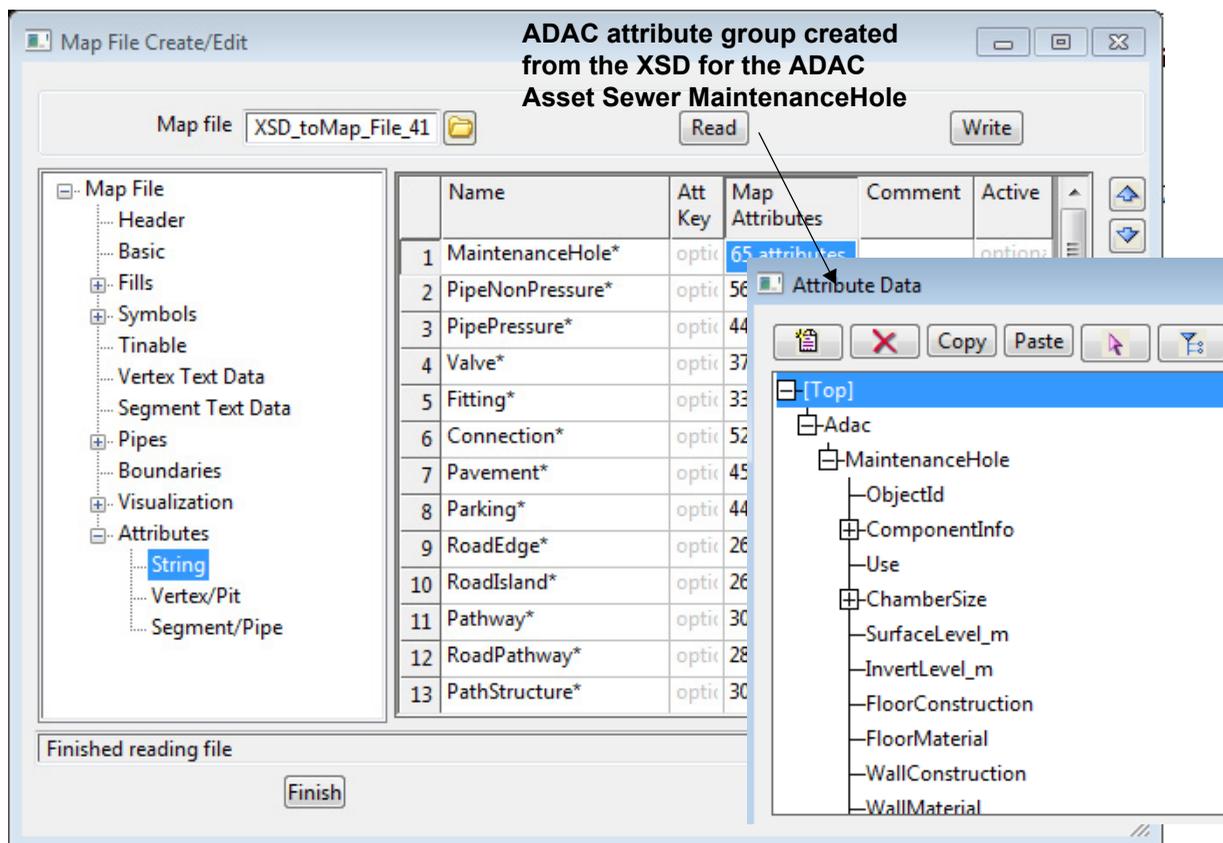
Continue to the next section [27.4.8.2 XSD to Map File](#) or return to [27.4.8 ADAC Utilities](#).

## 27.4.8.2 XSD to Map File

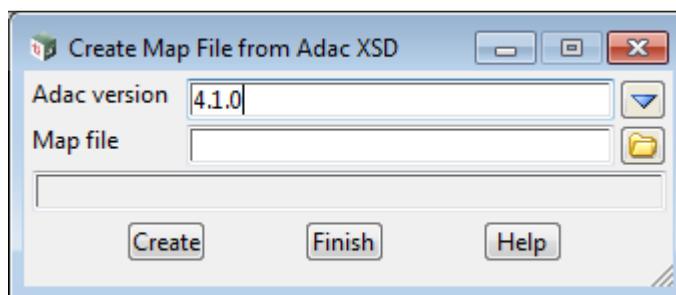
**Position of option on menu:** File I/O =>ADAC =>Utilities =>XSD to Map File

**XSD to Map File** takes the *ADAC XML Schema XSD* for a given version and creates a **12d Map File** (or just **Map file**), and in the *Attributes >String* section, creates a row for each Asset type in the XSD (sixty-eight of them) with the short name of the ADAC Asset followed by an \* in the **Name** column, and in the same row, creates an ADAC group of attributes for that Asset type in the **Map Attributes** column with default values for each of the ADAC elements in the ADAC group.

For example, running the **XSD to Map File** option for ADAC XML version 4.1.0 Schema, creates the **Map File**:



Selecting XSD to Map file brings up the **Create Map File from ADAC XSD** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>ADAC version</b> <i>this is the ADAC XML XSD to create the 12d Map File from.</i>	choice box	from env.4d	4.0.0, 4.1.0
<b>Map file</b> <i>the name of the <b>Map File</b> that is created. And in the <b>Attributes &gt;String</b> section, for each ADAC Asset in the XSD a row is created with the short name for the ADAC asset followed by an * in the <b>Name</b> column, and in the same row, an ADAC group is created in the <b>Map Attributes</b> column of the type, and all the elements in the XSD for that ADAC Asset are copied as attributes into the ADAC group.</i> <i>See the section <a href="#">9.8.1 Create/Edit a Map File</a> for information about 12d map files.</i>	file box		*.mapfile, *.mf files
<b>Create</b> <i>when clicked, the <b>Map File</b> is created.</i>	button		

Continue to the next section [27.4.8.3 XSD to Model](#) or return to [27.4.8 ADAC Utilities](#).

### 27.4.8.3 XSD to Model

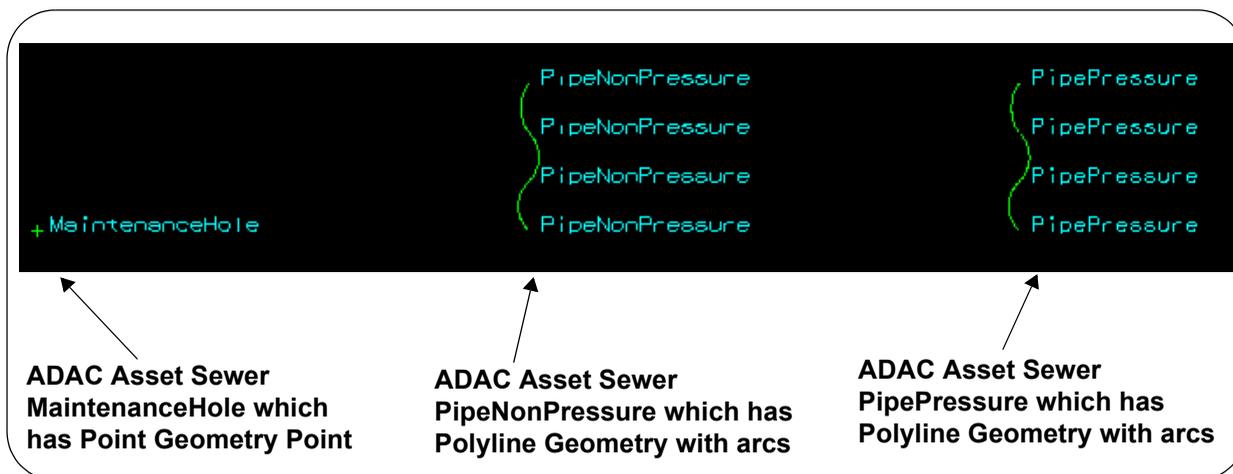
**Position of option on menu:** File I/O =>ADAC =>Utilities =>XSD to model

**XSD to Model** takes a given version of the *ADAC XML Schema* XSD and, for each ADAC Asset in the XSD, creates a super string using:

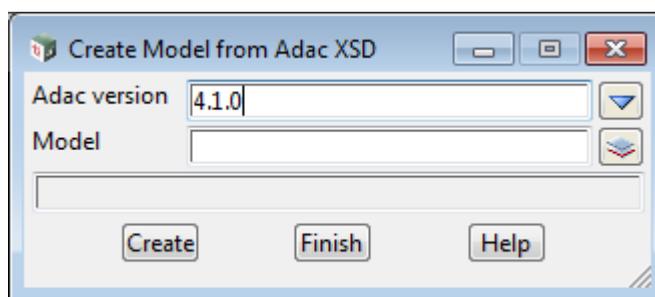
- (a) the short name of the ADAC Asset as the string name
- (b) in the string attributes, an ADAC group with default values for each of the ADAC elements
- (c) the ADAC Asset Geometry with default values as the string geometry
- (d) green for the string colour.

So there will be sixty-eight strings created for ADAC versions 4.1 or 4.0.

For example, running the **XSD to model** option for ADAC XML version 4.1.0 Schema creates a **model**, and the first three strings in that model are:



Selecting **XSD to model** brings up the **Create Model from ADAC XSD** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>ADAC version</b>	choice box	from env.4d	4.0.0, 4.1.0
<b>Model</b>	model box		available models

*this is the ADAC XML XSD to create the strings for:*

*for each ADAC Asset in the XSD, a super string is created using:*

- (a) the short name of the ADAC Asset as the string name
- (b) in the string attributes, an ADAC group with default values for each of the ADAC elements
- (c) the ADAC Asset Geometry with default values as the string geometry
- (d) green for the string colour.

*These ADAC strings are added to this model.*

*See the section [9.8.1 Create/Edit a Map File](#) for information about 12d map files.*

**Create** button

*when clicked, the model of strings for ADAC Assets in the XSD is created.*

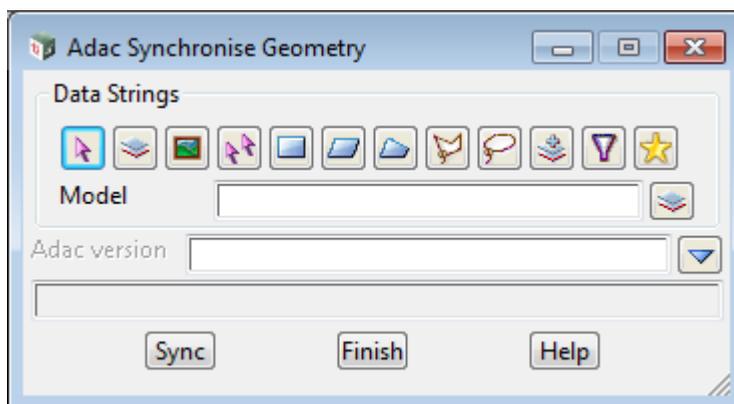
Continue to the next section [27.4.8.4 Sync Geometry](#) or return to [27.4.8 ADAC Utilities](#).

## 27.4.8.4 Sync Geometry

**Position of option on menu:** File I/O =>ADAC =>Utilities =>Sync geometry

**Sync geometry** takes selected strings representing ADAC Assets and updates the Geometry section of the ADAC attributes for the string with the actual geometry of the string.

Selecting **Sync geometry** brings up the **ADAC Synchronise Geometry** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to process. The ADAC Asset strings will have their ADAC Geometry attributes updated.*

**ADAC version**

choice box

from env.4d

4.0.0, 4.1.0

*this field is optional but if it is set, then only strings representing ADAC Assets of this version of the ADAC XML Schema will have their ADAC Geometry updated.*

**Sync**

button

*when clicked, the selected data will have its ADAC Geometry attributes updated from the actual string geometry.*

Continue to the next section [27.4.8.5 Create/Edit User Attributes to ADAC File](#) or return to [27.4.8 ADAC Utilities](#).

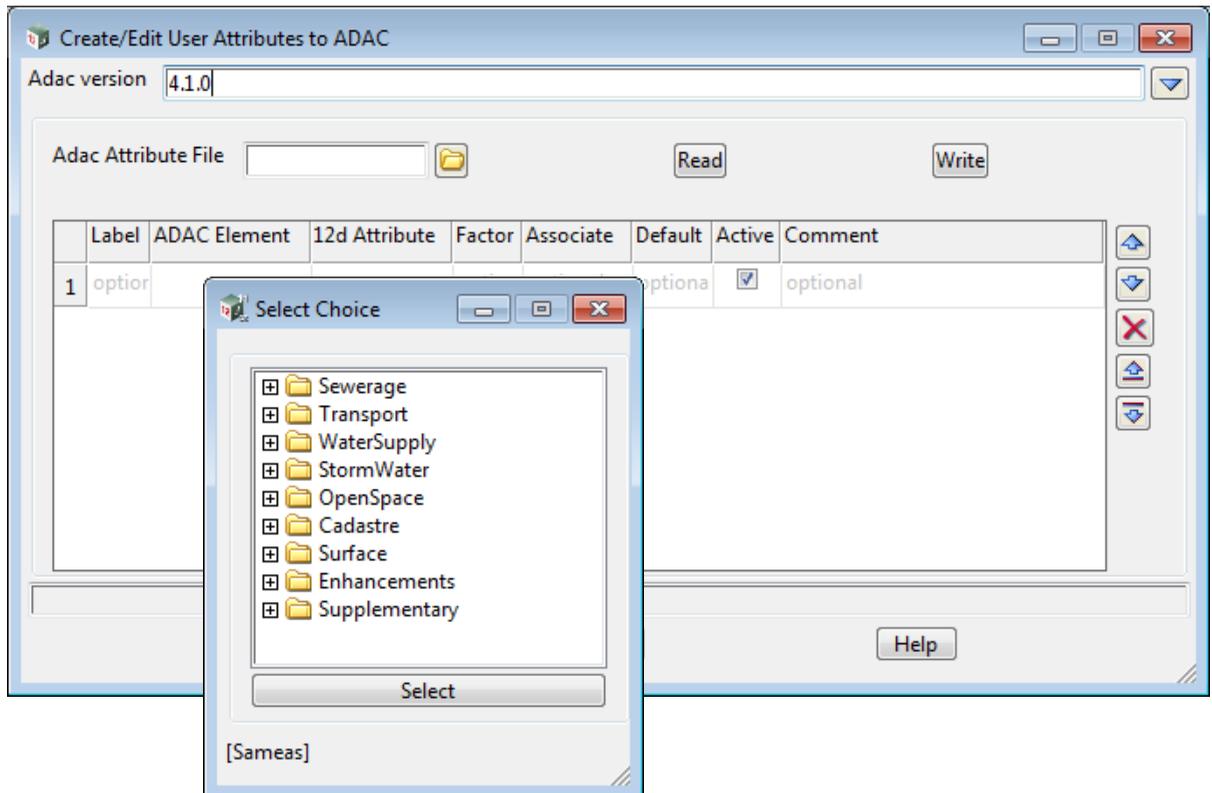
## 27.4.8.5 Create/Edit User Attributes to ADAC File

**Position of option on menu:** File I/O =>ADAC =>Utilities =>Create/Edit User Attributes to ADAC file

This option creates and edits a **12duaf** file which defines what 12d string, vertex or segment User attributes are used to update user specified ADAC Asset element values.

The **12duaf** file is used to update the values in ADAC Assets by the **Apply User Attributes to ADAC** panel (see [27.4.8.6 Apply User Attributes to ADAC Elements](#)).

Selecting **Create/Edit User Attributes to ADAC** file brings up the **Create/Edit User Attributes to ADAC** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>ADAC version</b>	choice box	from env.4d	4.0.0, 4.1.0

*the version of the ADAC XML Schema that the ADAC Asset elements are from. The version is necessary because the names can change between versions.*

<b>ADAC Attributes file</b>	file box		*.12duaf files
-----------------------------	----------	--	----------------

*name of the 12d User Attribute to ADAC file to read or write.*

<b>Read</b>	button
-------------	--------

*click on this button and the file given in the **ADAC Attribute file** field is read in and loaded into the editor.*

<b>Write</b>	button
--------------	--------

*writes out the data in the editor to the file given in the **ADAC Attribute file** field.*

### 12duaf File Grid

*when an 12duaf file is read in, it is displayed in the 12duaf tree.*

**Label** text

an optional unique name for this row of the grid. The name can only contain alphanumeric characters and underscores (\_).

The **Label** is for use in the **Associate** field of another row instead of using the full adac attribute path name.

**Note:** **Label** is only needed if the row is for a real or integer valued attribute and it is being referenced in an **Associate** column of another row. If this is not the case then simply leave **Label** blank.

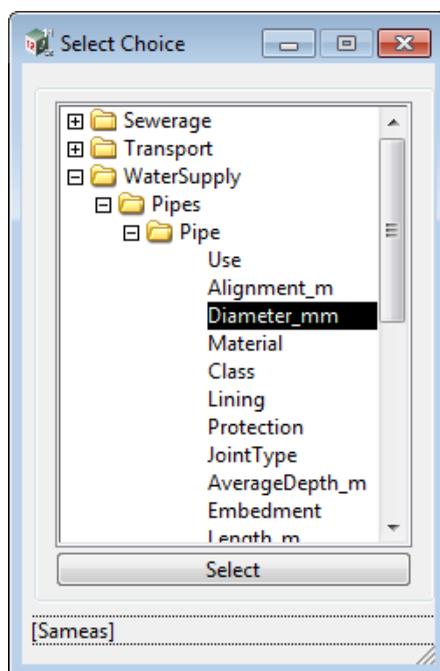
**ADAC Element** text column

the full name of the ADAC element to have its value updated and/or be used with **Label** in the **Associate** column of another row (see the documentation on the **Associate** column).

For example, the name could be:

*WaterSupply/Pipes/Pipe/Diameter\_mm*

**Important Note:** The name can be typed in but it is easiest to select it from the ADAC Schema by clicking RB in the **ADAC Element** row and expanding the tree and double clicking on the required ADAC element.



This ADAC element is given the value from the same row of the following **12d Attribute**, multiplied, when appropriate, by **Factor** when Factor is non zero.

**12d Attribute** text column

when **not blank**, the full path name of the **12d User Attribute** with

**String/** preceding the name if it is a **String Attribute**.

**Vertexn/** preceding the name if it is a **Vertex Attribute** on the **n**'th vertex.

**Segmentn/** preceding the name if it is a **Segment Attribute** on the **n**'th segment.

For example,

*Vertex1/Survey/Diameter*

is the Vertex attribute called **Survey/Diameter** on vertex 1

Note that **case** is important in **User Attribute** names. That is, upper and lower case characters are considered different.

When **blank**, the value in **Default** is used for the ADAC Element.

**Factor** real column

for a row where **Factor** is non zero, the value of the **ADAC Element** column is set equal to the value in

the **12d Attribute** column multiplied by the value in the **Factor** column.

For example, **Factor** would be set to 1000 if the **12d Attribute** was in metres and the **ADAC Element** was in millimetres.

### Associate real column

the **Associate** field allows arithmetic operations (addition +, subtraction -, multiplication \* and division /) and the 12d geometric functions:

*Height(integer\_val)* which give the height of vertex integer\_val

*Vertex\_count()* which gives the number of vertices in a string.

so *Height(Vertex\_count())* would be the height of the last vertex in a string.

If a value is required from a real or integer valued ADAC Element that is specified in another row, then that row is given a **Label** and the row is used in the formulae.

For example if we already have the surface level and the invert level for a StormWater Pit, the Depth (which is also needed) can be calculated from the other two values using the **Associate** field.

So you would have the following three rows in the 12duaf file - one with a Label SF\_Level and another with a Label IL\_Level:

**Label** SF\_Level for the row of the **ADAC Element** StormWater/Pits/Pit/SurfaceLevel\_m

**Label** IL\_Level for the row of the **ADAC Element** StormWater/Pits/Pit/InvertLevel\_m

Then to calculate the depth which is the difference of these two attributes, we create another row and use the **Associate** field to assign a value to the ADAC Element StormWater/Pits/Pit/Depth\_m that is the difference of the above labelled row.

So you add a new row which has in the **ADAC Element** and **Associate** fields:

**ADAC Element** field StormWater/Pits/Pit/Depth\_m

**Associate** field SF\_Level - IL\_Level

	Label	ADAC Element	12d Attribute	Factor	Associate	Default	Active	Comment
1	SF_Level	StormWater/Pits/Pit/SurfaceLevel_m		option	optional	optiona	<input checked="" type="checkbox"/>	optional
2	IL_Level	StormWater/Pits/Pit/InvertLevel_m		option	optional	optiona	<input checked="" type="checkbox"/>	optional
3	optional	StormWater/Pits/Pit/Depth_m		option	SF_Level - IL_Level	optiona	<input checked="" type="checkbox"/>	optional

### Default text column

this has three uses - giving the ADAC Element a value if the 12d Attribute doesn't exist or there is an error in the **Associate** column OR giving the ADAC Element a typed value OR deleting an ADAC Element.

(a) if the **12d Attribute** **doesn't exist** then the value in the **Default** field is used for the **ADAC Element**. Similarly if the row is using the **Associate** column and there is an error with it, then the value in the **Default** field is used for the **ADAC Element**.

(b) if the **12d Attribute** is **blank**, then the value in the **Default** field is used for the **ADAC Element**.

(c) if the text in the **Default** field is **-remove** then the **ADAC Element** will be **DELETED**. This can be useful when the user wants to deliberately make that ADAC Element invalid so that if someone doesn't add it in at a later time then it will show up as an error when running the **ADAC Validate** option.

#### **Important Note:**

For cases (a) and (b), the **Default** value **does not** have to be a **valid** value for the **ADAC Element**. This can be a method of making an ADAC Element invalid so that if someone doesn't put in a valid value at a later time then it will show up as an error when running the **ADAC Validate** option.

### Active

tick box

tick



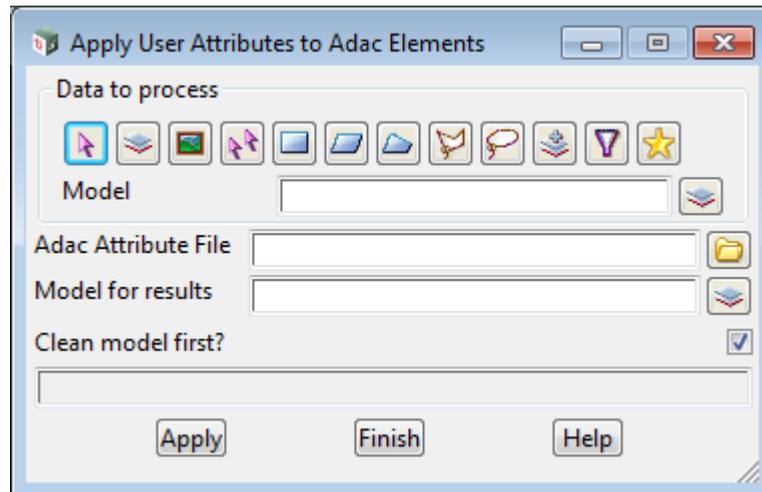
## 27.4.8.6 Apply User Attributes to ADAC Elements

**Position of option on menu:** File I/O =>ADAC =>Utilities =>Apply User Attributes to ADAC elements

This option applies a **12duaf** file to ADAC Assets to update the *ADAC elements* with values from the *12d User Attributes* of the ADAC Asset string.

The **12duaf** file is created and edited by the **Create/Edit User Attributes to ADAC** panel (see [27.4.8.5 Create/Edit User Attributes to ADAC File](#)).

Selecting **Apply User Attributes to ADAC elements** brings up the **Apply User Attributes to ADAC Elements** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of data to process. The ADAC Asset strings will have the ADAC Elements updated by User Attributes.</i>			
<b>ADAC Attributes file</b>	file box		*.12duaf files
<i>name of the 12d User Attribute to ADAC file to use.</i>			
<b>Model for results</b>	model box		available models
<i>model that the processed strings are added to.</i>			
<b>Clean model first ?</b>	tick box		
<i>if ticked, the model <b>Model for results</b> is cleaned before any strings are added to it. If not ticked, the model <b>Model for results</b> is NOT cleaned.</i>			
<b>Apply</b>	button		
<i>when clicked, the selected data will have any ADAC Elements updated by User Attributes according to the selected 12duaf file.</i>			

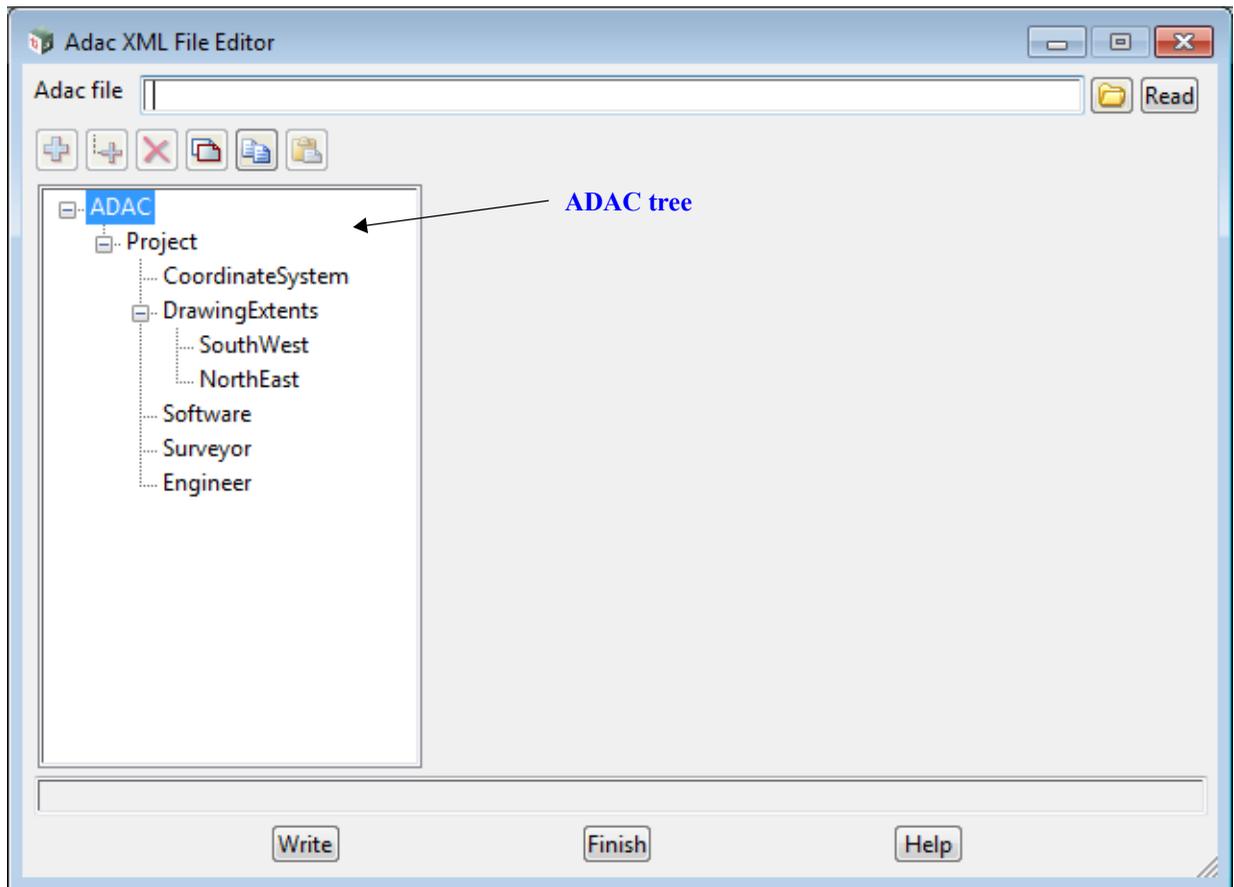
Continue to the next section [27.4.8.7 ADAC XML File Editor](#) or return to [27.4.8 ADAC Utilities](#).

## 27.4.8.7 ADAC XML File Editor

**Position of option on menu:** File I/O =>ADAC =>Utilities =>ADAC file editor

**ADAC XML file editor** reads an ADAC XML file and loads it into the **ADAC XML File Editor**.

Selecting **ADAC file editor** brings up the **ADAC XML File Editor** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>ADAC XML file</b>	file box		*.XML files
<i>name of the XML file to read or write.</i>			

**Read** button

*click on this button and the file given in the **ADAC XML file** field is read in and loaded into the editor.*

### ADAC tree

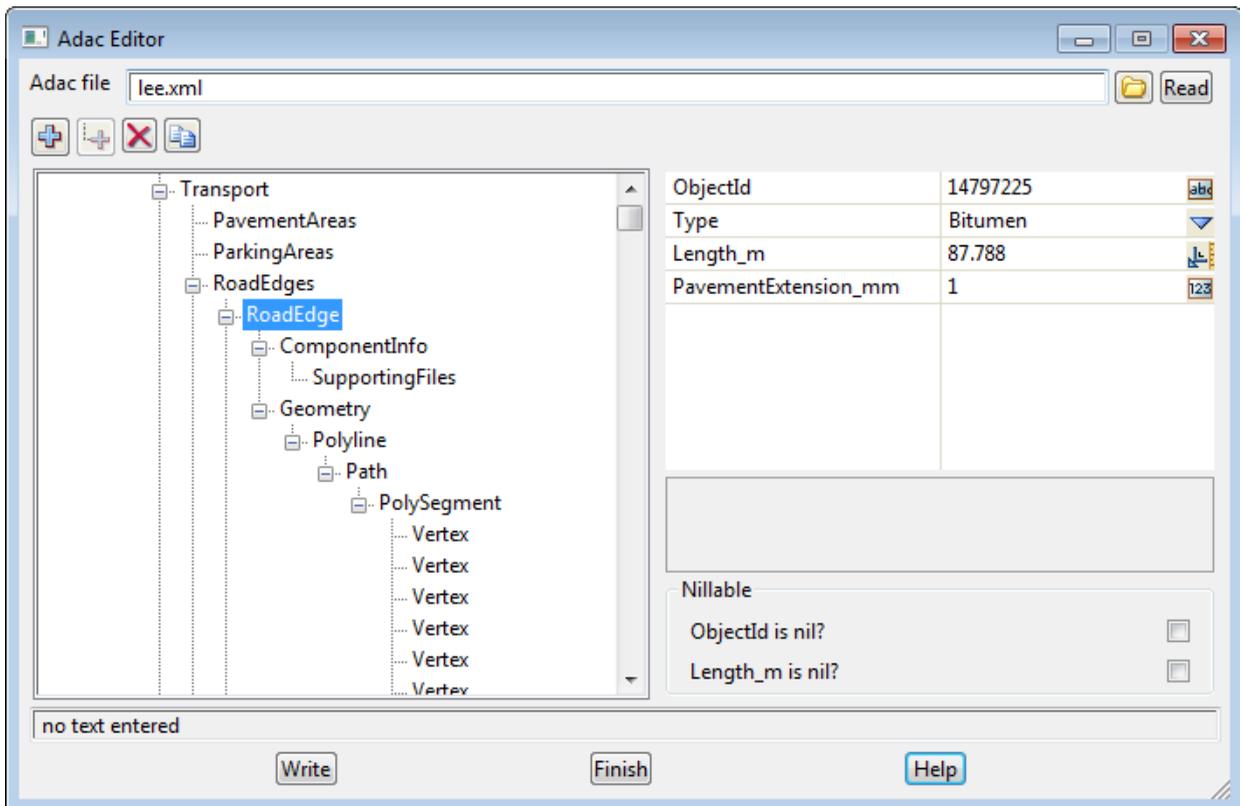
*when an ADAC XML file is read in, it is displayed in the ADAC tree.*

*Elements in the tree can be edited according to the ADAC XSD. New ADAC Assets can also be added.*

**Write** button

*writes out the data in the editor to the file given in the **ADAC XML file** field.*

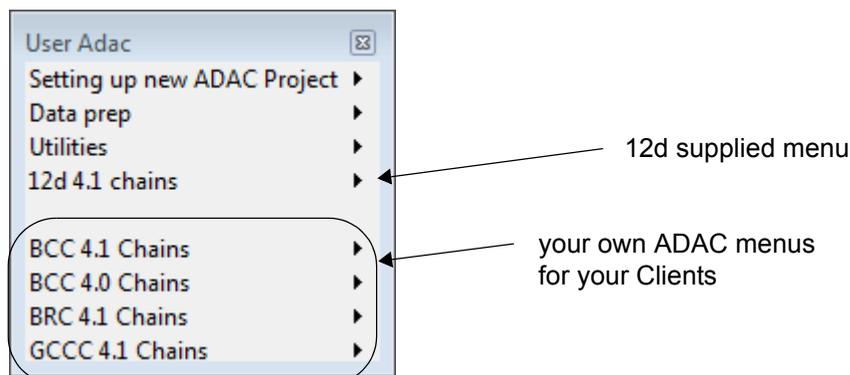
As an example,



Return to [27.4.8 ADAC Utilities](#).

## 27.4.9 User ADAC

**Position of menu:** File I/O =>ADAC =>User



See

[27.4.9.1 Setting Up a New ADAC Project](#)

[27.4.9.2 Data Prep](#)

[27.4.9.3 User ADAC Utilities](#)

The easiest way to run the ADAC *Survey* or *Design* chains with specific pvf files is to run them from a menu. This can be done in two ways:

- (a) Running Specific Map and 12duaf files from Local and User\_Lib

**12d Model** provides options on the walk right menu **Run 12d ADAC 4.1 Chains** on the option **12d 4.1 chains** to run ADAC *Survey* or *Design* chains for ADAC 4.1 with specially named Map and 12duaf files in the working folder for the project (local), or User\_Lib or Library.

See [27.4.9.4 12d 4.1 Chains](#).

- (b) Running Client Specific Map and 12duaf files

Once you are producing ADAC files for different Clients, you may need to produce different ADAC versions, or need different Map or 12duaf files.

For example you may need to use a different naming convention, or the information required in the ADAC assets is different.

For these situations, you can add your own options to **User Adac** menu.

The user options can run either the ADAC Base Survey or Design chains with Client specific chain pvf files.

For information on setting up your own options on the **User ADAC** menu, see [27.4.9.5 Client Specific ADAC Design & Survey Menus](#).

Or return to [27.4.8 ADAC Utilities](#).

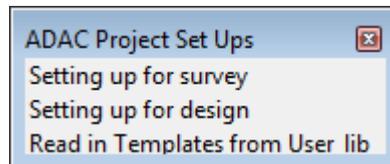
## 27.4.9.1 Setting Up a New ADAC Project

**Position of menu:** File I/O =>ADAC =>User =>Setting up a new ADAC Project

A new project for producing ADAC XML needs to be set up a certain way for the ADAC Chains to work.

There is a different setup for a **Survey** Project to that for a **Design** Project.

ADAC Header and Asset templates can also be read in from the file **ADAC\_Templates.12da** in User\_Lib.



See

[27.4.9.1.1 Setting Up for Survey](#)

[27.4.9.1.2 Setting Up for Design](#)

[27.4.9.1.3 Reading in Templates from User\\_Lib](#)

### 27.4.9.1.1 Setting Up for Survey

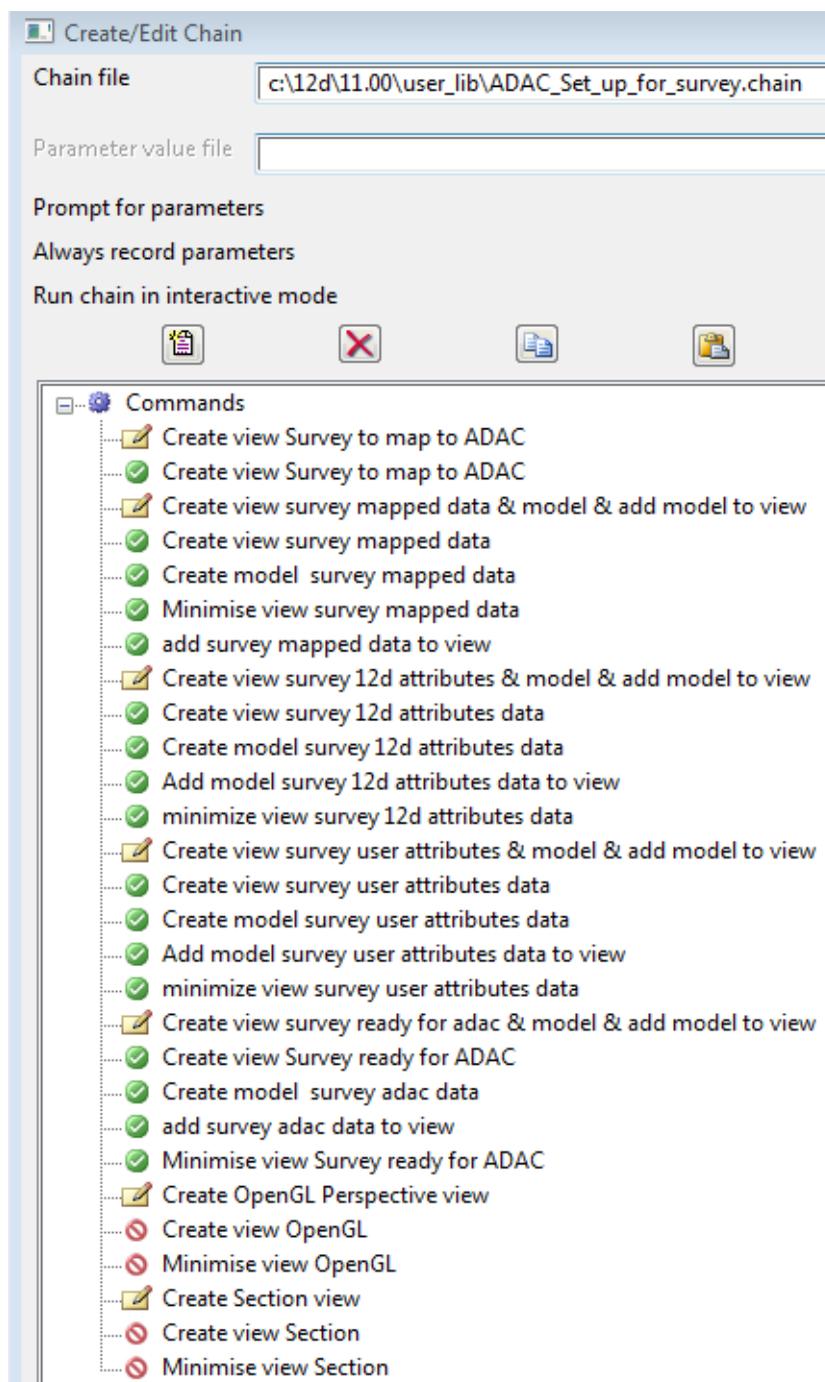
Position of option on menu:

File I/O =>ADAC =>User =>Setting up a new ADAC Project =>Setting up for survey

**Setting up for survey** runs the Chain *ADAC\_Set\_up\_for\_survey.chain*, which is in *\$LIB*.

This chain creates some new Plan views and some models which are added to the views, and the views then minimised.

The Plan views and models are used by the ADAC Survey Chain.



Continue to the next section [27.4.9.1.2 Setting Up for Design](#) or return to [27.4.9.1 Setting Up a New ADAC Project](#).

### 27.4.9.1.2 Setting Up for Design

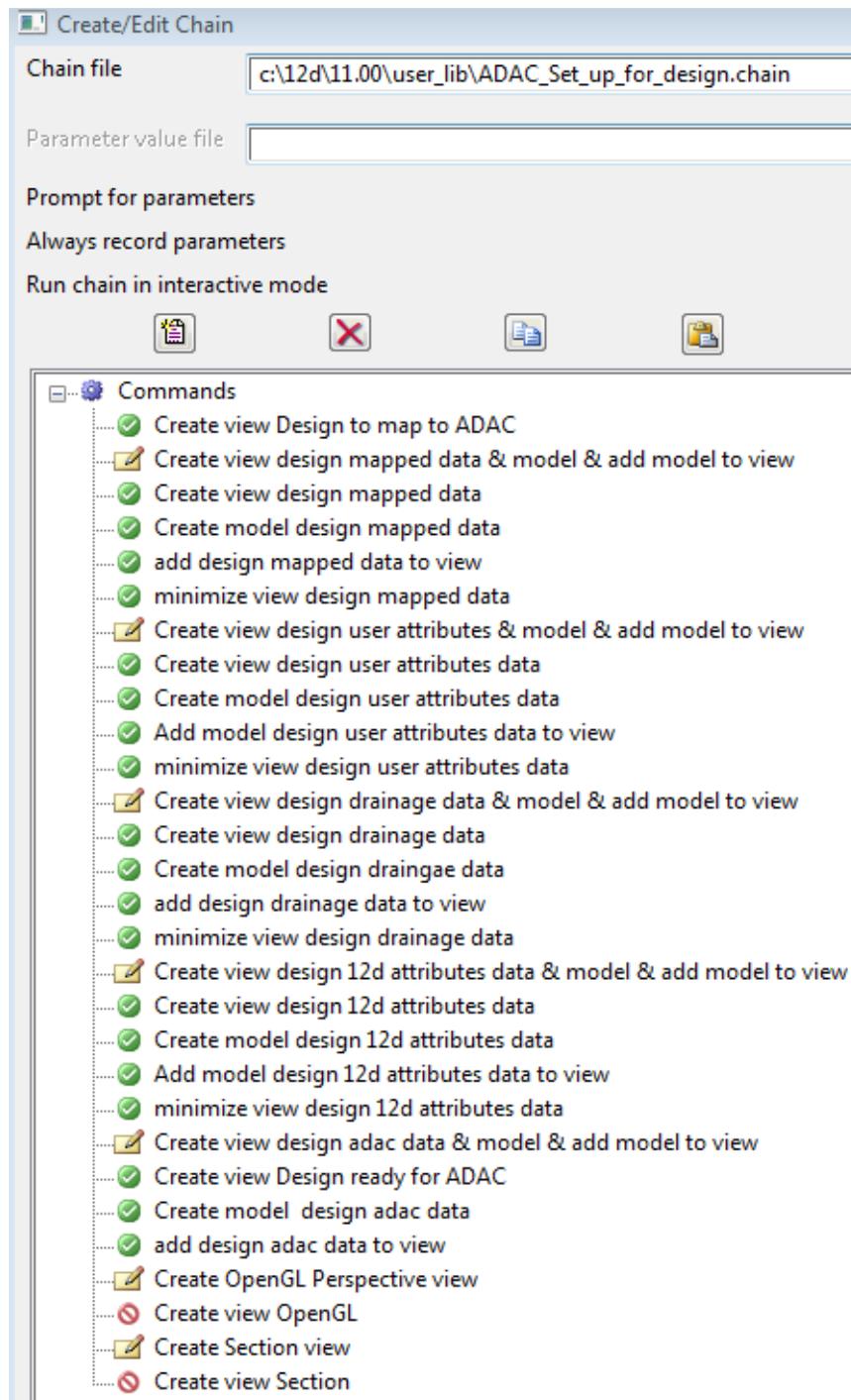
Position of option on menu:

File I/O =>ADAC =>User =>Setting up a new ADAC Project =>Setting up for design

**Setting up for design** runs the Chain **ADAC\_Set\_up\_for\_design.chain** which is in **\$LIB**

This chain creates some new Plan views and some models which are added to the views, and the views then minimised.

The Plan views and models are used by the ADAC Design Chain.



Continue to the next section [27.4.9.1.3 Reading in Templates from User\\_Lib](#) or return to [27.4.9.1 Setting Up a New ADAC Project](#).

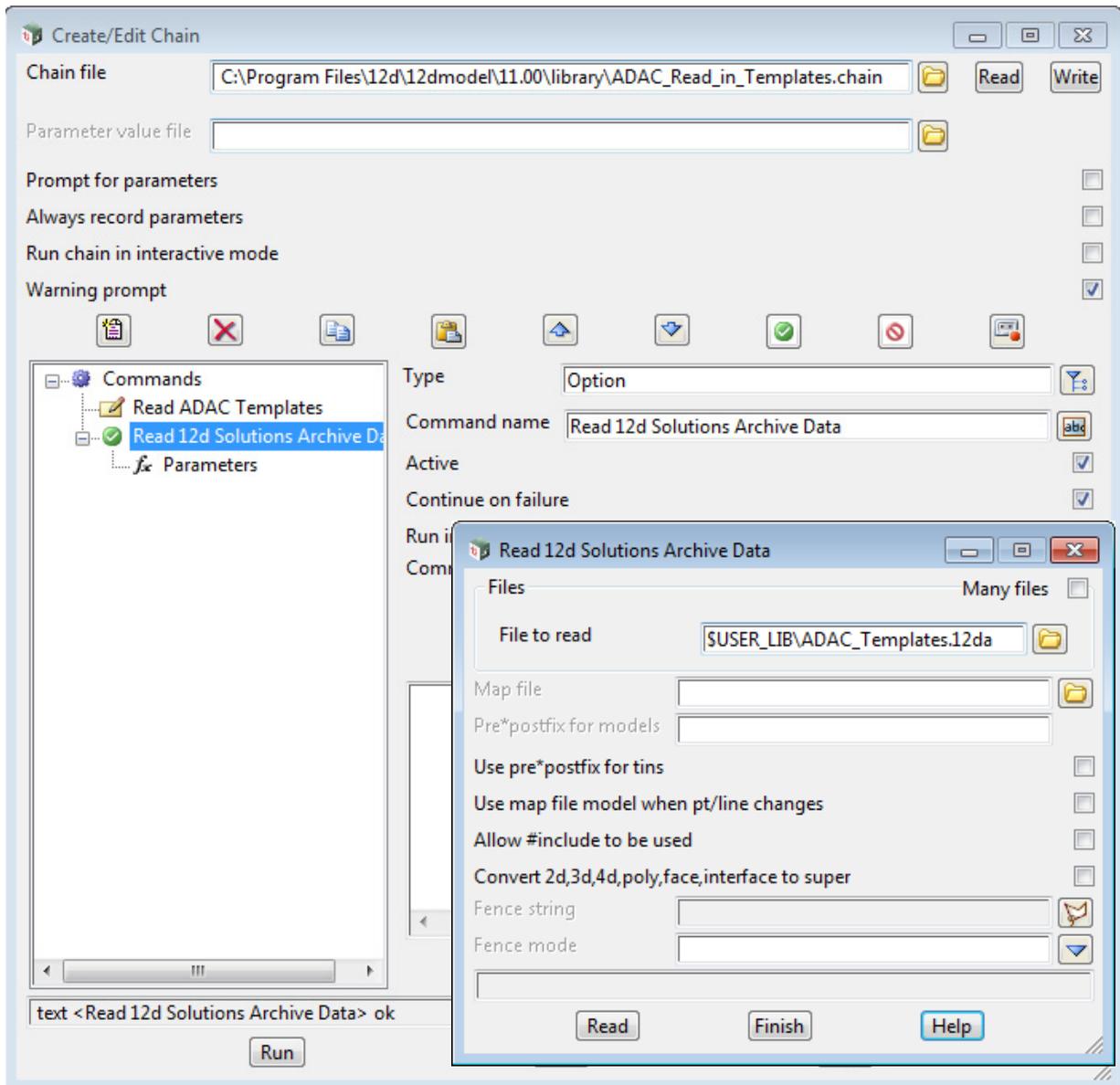
### 27.4.9.1.3 Reading in Templates from User\_Lib

Position of option on menu:

File I/O =>ADAC =>User =>Setting up a new ADAC Project =>Read in templates from User\_Lib

**Reading Templates from User\_Lib** runs the Chain *ADAC\_Read\_in\_Templates.chain* which is in *Library*.

This chain reads in the file *ADAC\_Templates.4da* from *User\_Lib*.



So if you have created Header and Assets Templates then by writing the models **ADAC Header Templates** and **ADAC Asset Templates** which contain the Templates out to this then this option can be used to read them into a new project.

For information on ADAC Header and Asset Templates, see [27.6.4 Setting Up ADAC Templates](#).

Return to [27.4.9.1 Setting Up a New ADAC Project](#).

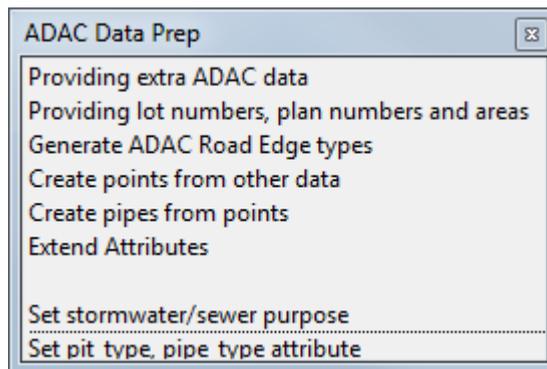
## 27.4.9.2 Data Prep

**Position of menu:** File I/O =>ADAC =>User =>Data prep

When you start with a Survey and Design in **12d Model**, not all the information required for ADAC will necessarily be there. In fact, some of the ADAC information may even be added at a later stage by someone other than the Surveyor or Designer.

Similarly, some of the strings may not be of the correct type to become an ADAC Asset. For example, a string may not have the correct geometry as defined for the ADAC Asset in the *ADAC XML Schema XSD*.

So before running an ADAC Survey or ADAC Design chain, one or more of the **ADAC Data Prep** options may need to be run to prepare the strings that will become ADAC Assets.



See

[27.4.9.2.1 Providing Extra Data for ADAC](#)

[27.4.9.2.2 Providing Lot and Plan Numbers and Areas for ADAC](#)

[27.4.9.2.3 Generate ADAC Road Edge Types](#)

[27.4.9.2.4 Create Points from Other Data](#)

[27.4.9.2.5 Create Pipes from Points](#)

[27.4.9.2.6 Extend Attributes](#)

[27.4.9.2.7 Set Stormwater/Sewer Property](#)

[27.4.9.2.8 Set a Drainage-Sewer Pit and Pipe Type Attribute](#)

[27.4.9.2.9 Move/Link Pipe Ends to Pit Centres](#)

### 27.4.9.2.1 Providing Extra Data for ADAC

**Position of option on menu:** File I/O =>ADAC =>User =>Data prep =>Providing extra ADAC data

This option creates, as string attributes, information that is required for ADAC but is not normally part of a survey or a design.

The extra information is stored as attributes with the string, but not as ADAC attributes. A macro in the *ADAC Survey* or *ADAC Design* chain moves them into ADAC attributes. See ([27.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey](#))

A major reason not to create ADAC attributes straightaway is that although this option allows you to enter the values, in the future the values might already exist in another form and routines written to automatically get them. In this case it will be much easier for anyone to put them into a simpler attribute structure and not have to know anything about the complex ADAC attribute structure.

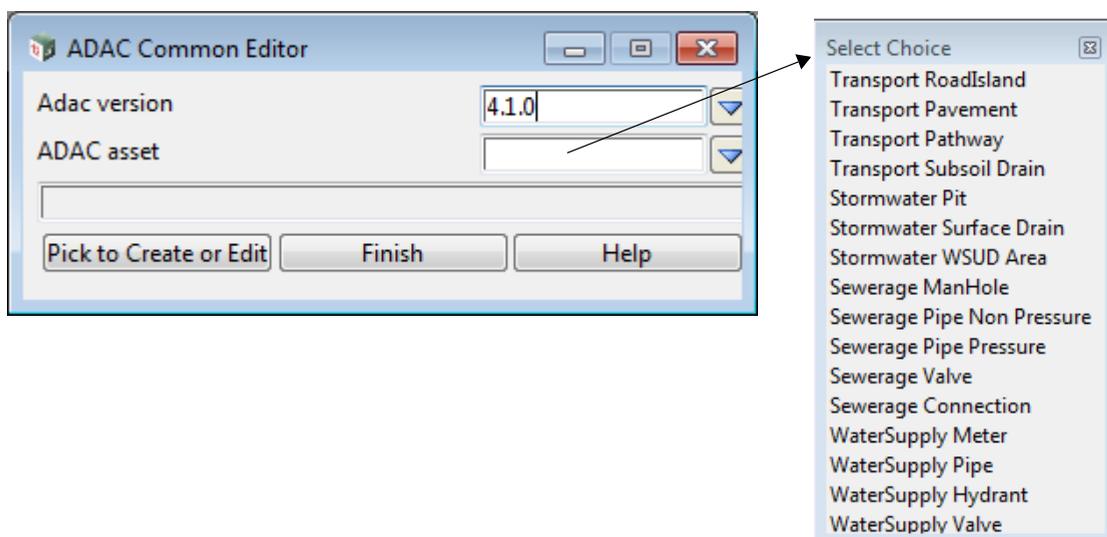
Another reason is that at this stage the *ADAC Survey* and *Design* chains have not been run to mark strings as ADAC Assets and set up the ADAC attributes group.

And when the *ADAC Survey* and *Design* chains are run, at each step in the chain they work with copies of the original data and clean out the models from a previous run rather than updating the original data. This is for safety so that it is easy to see that each step in the chain has worked, and if there is a problem, the chains can be run again and again.

So placing the attributes on the original data ensures they will still be there when the *ADAC Survey* or *Design* chains are rerun.

One important thing to keep in mind when running this option is that although you are using it to set up attributes that will go to ADAC, when you are picking the strings they are not yet marked as ADAC Assets. So you have to know **what ADAC Asset** the string is **going to become**.

Selecting **Providing extra data for ADAC** brings up the **ADAC Common Editor** panel



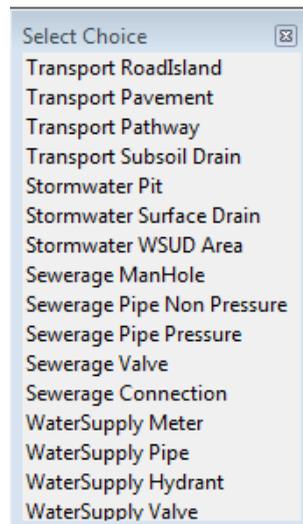
The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>ADAC version</b>	choice box	from env.4d	4.0.0, 4.1.0

*because the strings to be picked have not yet been marked as ADAC Assets, you need to let the option know what the ADAC version is going to be. This is required so that the correct ADAC Schema XSD is used to get information for the choices.*

ADAC asset

choice box



*the ADAC Asset that this string will become.*

**Pick to Create or Edit** button

*click on the button and then pick a string.*

*If the picked string already has attributes set by this option then those values will be displayed and can be edited.*

*If the picked string has not yet had attributes set by this option AND **ADAC asset** has a value, then a panel with the attributes that this option can set for the given **ADAC asset** is displayed and then created for the picked string.*

[27.4.9.2.1.1 Transport RoadIsland](#)

[27.4.9.2.1.2 Transport Pavement](#)

[27.4.9.2.1.3 Transport Pathway](#)

[27.4.9.2.1.4 Transport Subsoil Drain](#)

[27.4.9.2.1.5 Stormwater Pit](#)

[27.4.9.2.1.6 Stormwater Surface Drain](#)

[27.4.9.2.1.7 Stormwater WSUD Area](#)

[27.4.9.2.1.8 Sewerage Maintenance Hole](#)

[27.4.9.2.1.9 Sewerage Pipe Non Pressure](#)

[27.4.9.2.1.10 Sewerage Pipe Pressure Editor](#)

[27.4.9.2.1.11 Sewerage Valve Editor](#)

[27.4.9.2.1.12 Sewerage Connection Editor](#)

[27.4.9.2.1.13 Water Supply Meter Editor](#)

[27.4.9.2.1.14 Water Supply Pipe Editor](#)

[27.4.9.2.1.15 Water Supply Hydrant Editor](#)

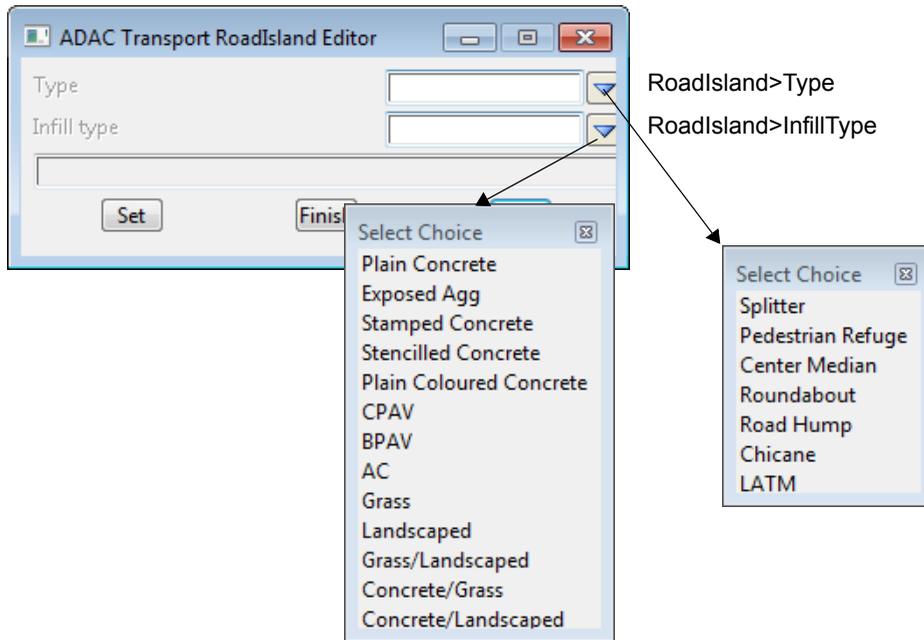
[27.4.9.2.1.16 Water Supply Valve Editor](#)

### 27.4.9.2.1.1 Transport RoadIsland

The choice of **Transport RoadIsland** brings up a panel for setting parameters that will become the values for the ADAC attributes **Type** and **InfillType** for the ADAC Asset **Transport>RoadIslands>RoadIsland**.

Because the option knows the *ADAC Schema* version and the *ADAC Asset*, the choices for **Type** and **InfillType** come from the *ADAC XSD*.

The selected string must be a Polygon with straight or arc segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

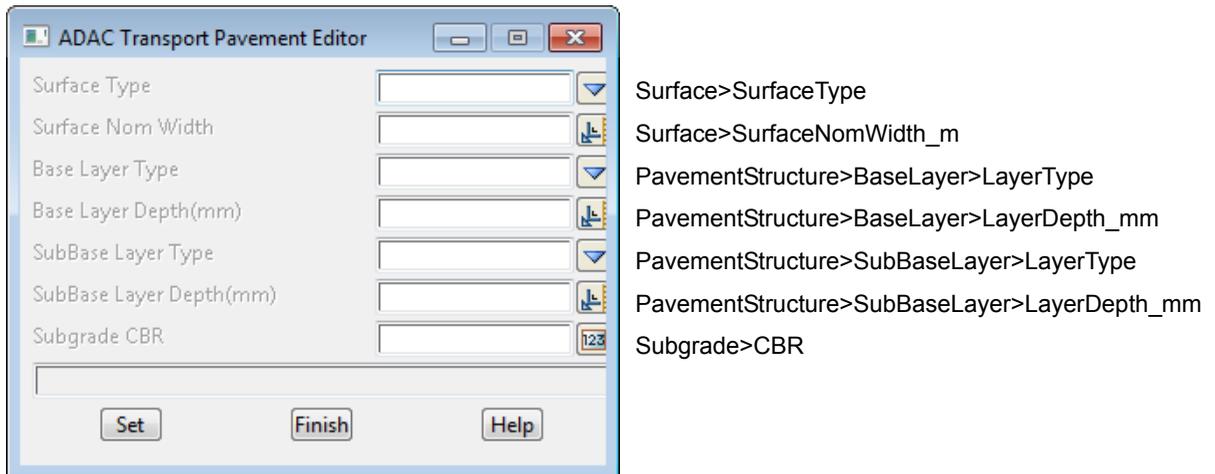
Continue to the next section [27.4.9.2.1.2 Transport Pavement](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.2 Transport Pavement

The choice of **Transport Pavement** brings up a panel for setting some of the parameters for the ADAC Asset **Transport>Pavement Areas>Pavement**.

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a Polygon with straight or arc segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

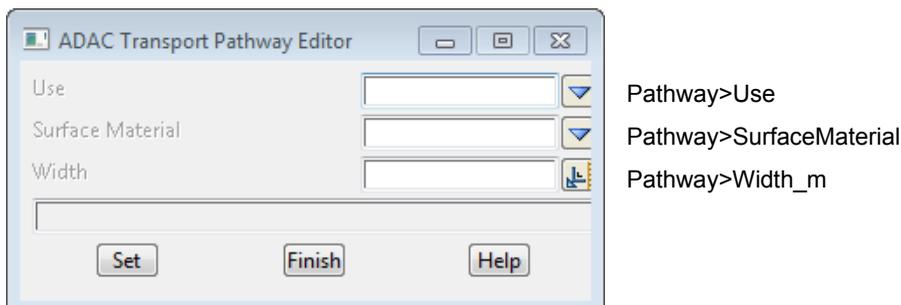
Continue to the next section [27.4.9.2.1.3 Transport Pathway](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.3 Transport Pathway

The choice of **Transport Pathway** brings up a panel for setting some of the parameters for the ADAC Asset **Transport>Pathways>Pathway**.

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a Polyline with straight or arc segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

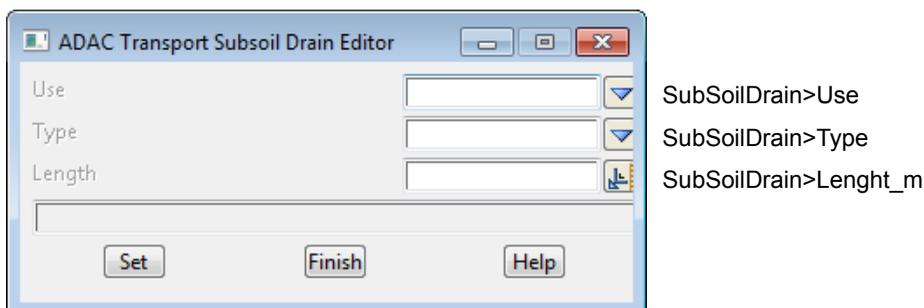
Continue to the next section [27.4.9.2.1.4 Transport Subsoil Drain](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.4 Transport Subsoil Drain

The choice of **Transport Subsoil Drain** brings up a panel for setting some of the parameters for the ADAC Asset **Transport>SubSoilDrains>SubSoilDrain**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a Polyline with only straight segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

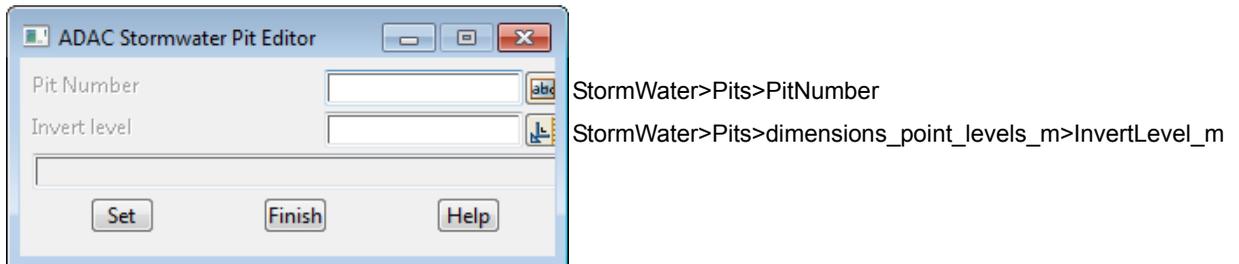
Continue to the next section [27.4.9.2.1.5 Stormwater Pit](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.5 Stormwater Pit

The choice of **Stormwater Pit** brings up a panel for setting some of the parameters for the ADAC Asset **StormWater>Pits>Pit**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a one vertex string.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

Continue to the next section [27.4.9.2.1.6 Stormwater Surface Drain](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.6 Stormwater Surface Drain

The choice of **Stormwater Surface Drain** brings up a panel for setting some of the parameters for the ADAC Asset **StormWater>SurfaceDrains>SurfaceDrain**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a Polyline with only straight segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

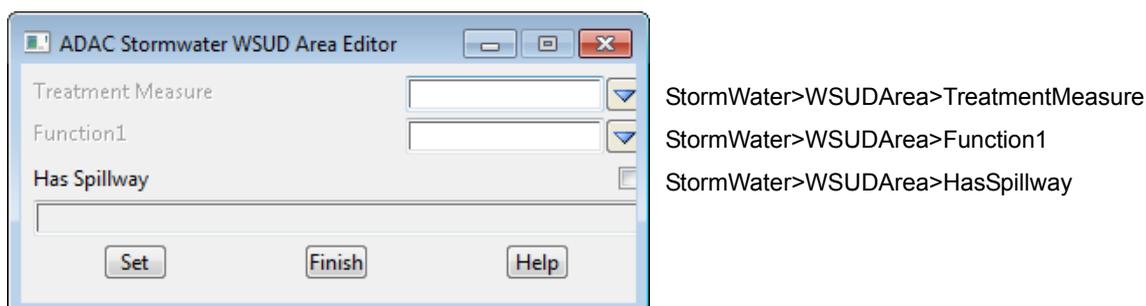
Continue to the next section [27.4.9.2.1.7 Stormwater WSUD Area](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.7 Stormwater WSUD Area

The choice of **Stormwater WSUD Area** brings up a panel for setting some of the parameters for the ADAC Asset **StormWater>WSUDAreas>WSUDArea**.

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a Polyline with straight or arc segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

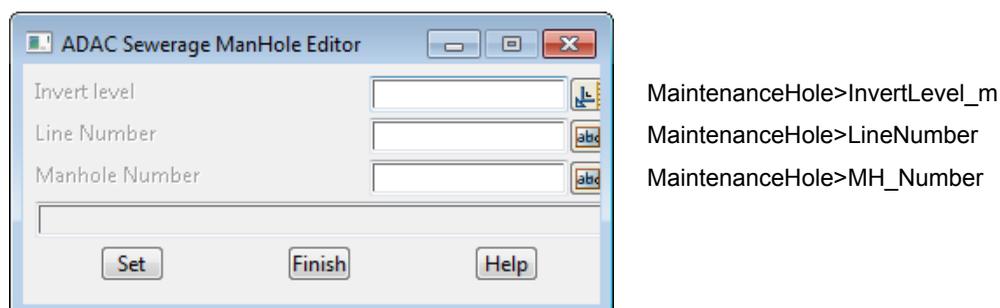
Continue to the next section [27.4.9.2.1.8 Sewerage Maintenance Hole](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.8 Sewerage Maintenance Hole

The choice of **Sewerage Maintenance Hole** brings up a panel for setting some of the parameters for the ADAC Asset **Sewerage>MaintenanceHoles>MaintenanceHole**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a one vertex string.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

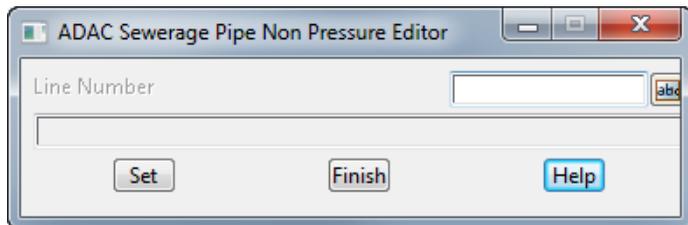
Continue to the next section [27.4.9.2.1.9 Sewerage Pipe Non Pressure](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.9 Sewerage Pipe Non Pressure

The choice of **Sewerage Pipe Non Pressure** brings up a panel for setting some of the parameters for the ADAC Asset **Sewerage>PipesNonPressure>PipeNonPressure**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a polyline, including arcs.



PipeNonPressure >LineNumber

**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

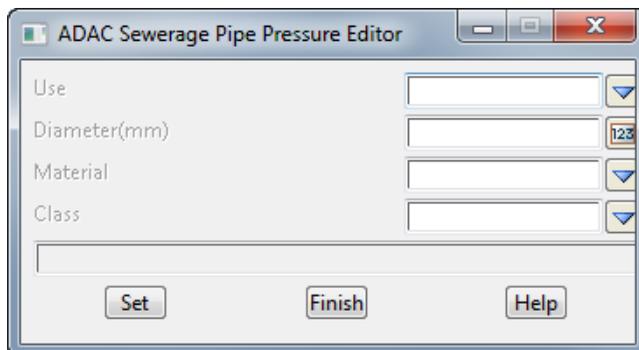
Continue to the next section [27.4.9.2.1.10 Sewerage Pipe Pressure Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.10 Sewerage Pipe Pressure Editor

The choice of **Sewerage Pipe Pressure** brings up a panel for setting some of the parameters for the ADAC Asset **Sewerage>PipesPressure>PipePressure**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a polyline which may include arcs.



PipePressure>Use

PipePressure>Diameter\_mm

PipePressure>Material

PipePressure>Class

**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

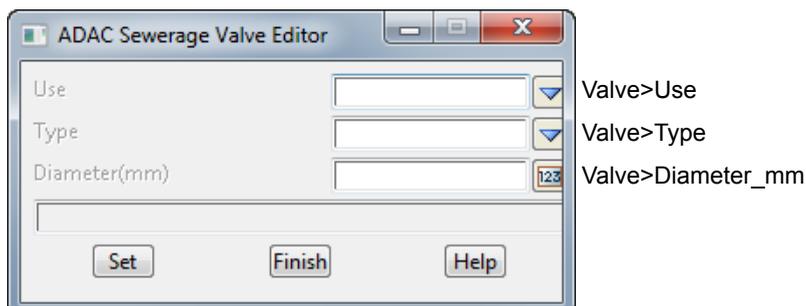
Continue to the next section [27.4.9.2.1.11 Sewerage Valve Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.11 Sewerage Valve Editor

The choice of **Sewerage Valve** brings up a panel for setting some of the parameters for the ADAC Asset **Sewerage>Valves>Valve**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a one vertex string.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

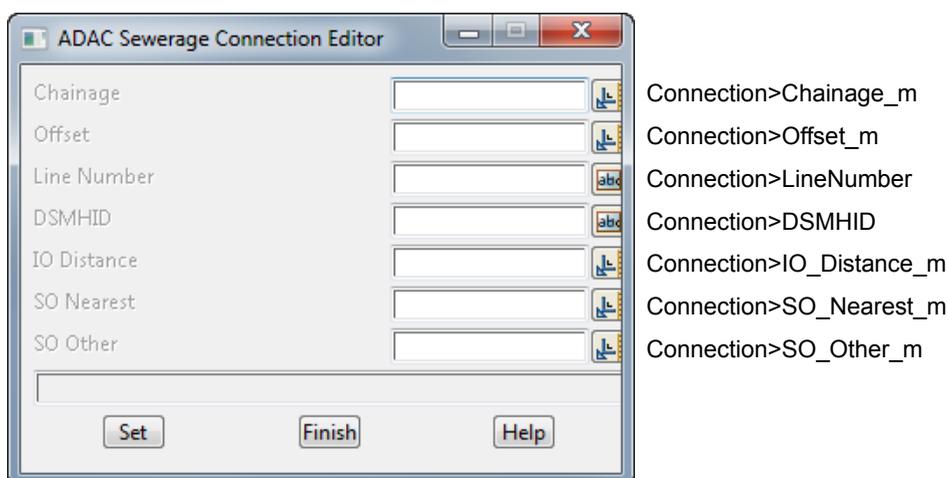
Continue to the next section [27.4.9.2.1.12 Sewerage Connection Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.12 Sewerage Connection Editor

The choice of **Sewerage Connection** brings up a panel for setting some of the parameters for the ADAC Asset **Sewerage>Connections>Connection**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a polyline which may include arcs.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

Continue to the next section [27.4.9.2.1.13 Water Supply Meter Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.13 Water Supply Meter Editor

The choice of **WaterSupply Meter** brings up a panel for setting some of the parameters for the ADAC Asset **WaterSupply>Meters>Meter**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a one vertex string.

Meter>SerialNumber

Meter>InitialReading

Meter>OffsetSide

Meter>Offset\_m

Meter>Group: lot\_plan\_details>LotNo

Meter>Group: lot\_plan\_details>PlanNo

**Pick lot** - after clicking on **Pick lot**, if a polygon is selected that has Lot and Plan attributes set using the **Provide lot numbers, plan numbers and areas** option (see [27.4.9.2.2 Providing Lot and Plan Numbers and Areas for ADAC](#)), then the values for LotNo and PlanNo are written into the **LotNo** and **PlanNo** fields.

**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

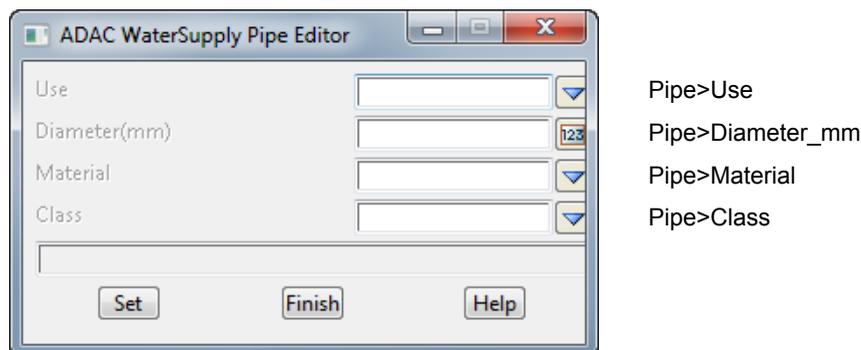
Continue to the next section [27.4.9.2.1.14 Water Supply Pipe Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.14 Water Supply Pipe Editor

The choice of **WaterSupply Pipe** brings up a panel for setting some of the parameters for the ADAC Asset **WaterSupply>Pipes>Pipe**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a polyline but only with straight segments.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

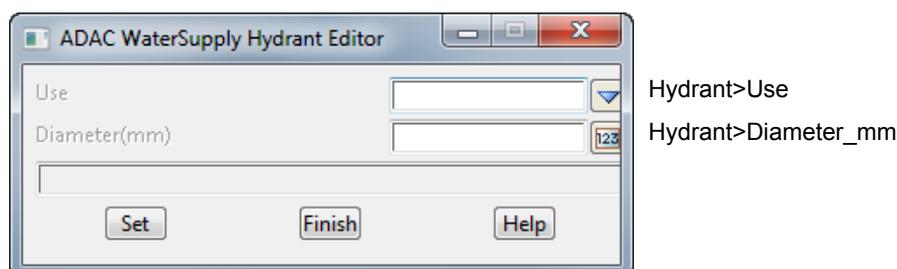
Continue to the next section [27.4.9.2.1.15 Water Supply Hydrant Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.1.15 Water Supply Hydrant Editor

The choice of **WaterSupply Hydrant** brings up a panel for setting some of the parameters for the ADAC Asset **WaterSupply>Hydrants>Hydrant**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a one vertex string.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

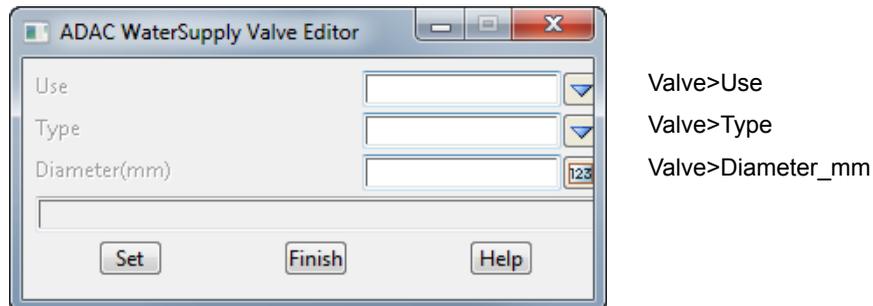
Continue to the next section [27.4.9.2.1.16 Water Supply Valve Editor](#) or return to [27.4.9.2.1 Providing Extra Data for ADAC](#).

### 27.4.9.2.16 Water Supply Valve Editor

The choice of **WaterSupply Valve** brings up a panel for setting some of the parameters for the ADAC Asset **WaterSupply>Valves>Valve**

Because the option knows the ADAC version and the ADAC Asset, the types for the attributes and any pop-up choices all come from the ADAC XSD.

The selected string must be a one vertex string.



**Set** writes the values in the panel as string attributes.

Clicking **Finish** removes the panel and replaces it with the ADAC Common Editor panel so another value of ADAC Asset can be chosen.

Continue to the next section [27.4.9.2.2 Providing Lot and Plan Numbers and Areas for ADAC](#) or return to [27.4.9.2 Data Prep](#).

## 27.4.9.2.2 Providing Lot and Plan Numbers and Areas for ADAC

### Position of menu:

File I/O =>ADAC =>User =>Data prep =>Providing lot numbers, plan numbers and areas

This option creates as string attributes: Plan numbers, Lot numbers and areas of lots. The area is calculated from the lot polygon, but that can be manually changed. However, having the actual area there to start with usually means that only the last couple of digits need to be entered.

The extra information is stored as attributes with the string, but not as ADAC attributes. A macro in the *ADAC Survey* or *ADAC Design* chain moves them into ADAC attributes. See ([27.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey](#))

A major reason not to create ADAC attributes straightaway is that although this option allows you to enter the values, in the future the values might already exist in another form and routines written to automatically get them. In this case it will be much easier for anyone to put them into a simpler attribute structure and not have to know anything about the complex ADAC attribute structure.

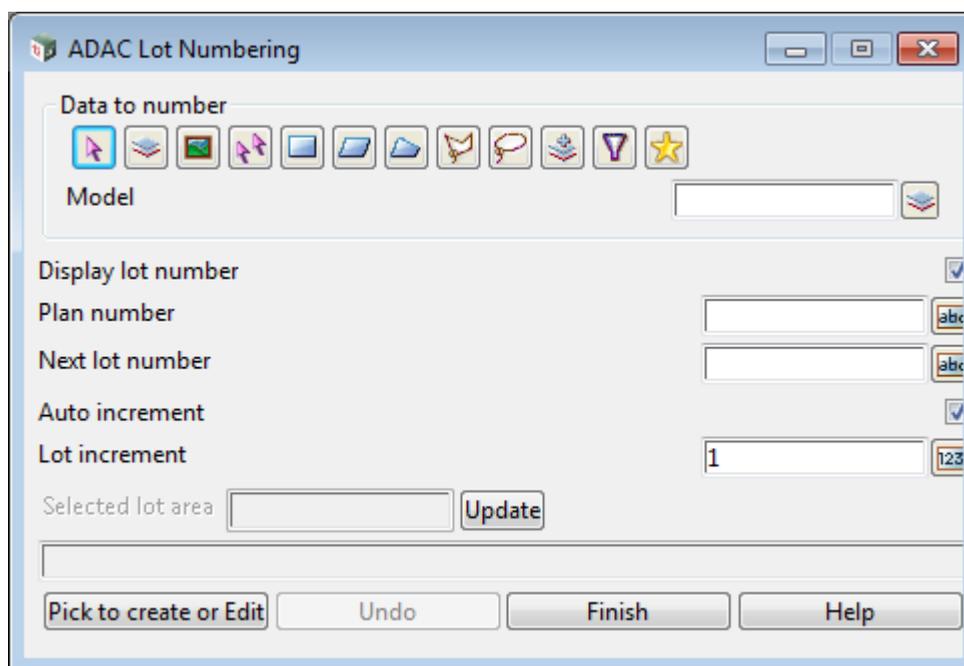
Another reason is that at this stage the *ADAC Survey* and *Design* chains have not been run to mark strings as ADAC Assets and set up the ADAC attributes group.

Also, when the *ADAC Survey* and *Design* chains are run, at each step in the chain they work with copies of the original data and clean out the models from a previous run rather than updating the original data. This is for safety so that it is easy to see that each step in the chain has worked, and if there is a problem, the chains can be run again and again.

So placing the attributes on the original data ensures they will still be there when the *ADAC Survey* or *Design* chain is rerun.

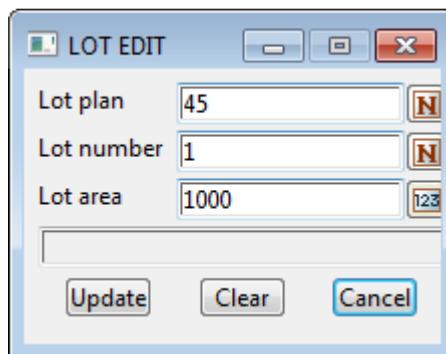
One important thing to keep in mind when running this option is that although you are using it to set up Lot attributes that will go to ADAC, when you are picking the strings they are not yet marked as ADAC *Cadastre>LandParcels>Lot* Assets. So you have to know **what ADAC Asset** that the string is **going to become** in ADAC *Cadastre>LandParcels>Lot*.

Selecting **Providing lot number, plan numbers and areas** brings up the **Lot Numbering and Areas** panel:



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of data that is be searched for any existing lot numbers and if they exist, the minimum and maximum existing lot numbers are written to the panel's message area and the highest lot number plus the <b>Lot increment</b> is written to the <b>Next lot no.</b> field.</i>			
<i>If a Model is selected by MB same as, you need to type &lt;Enter&gt; after it for the search to go ahead.</i>			
<b>Display lot number</b>	tick box	tick	
<i>if <b>ticked</b>, the lot numbers for existing lots are temporarily displayed on the view.</i>			
<i>If <b>not ticked</b>, the lot numbers are not temporarily displayed on the view.</i>			
<b>Plan number</b>	text box		
<i>number of the plan the lots are from.</i>			
<b>Next lot number</b>	text box		
<i>the next lot number to use.</i>			
<b>Auto increment</b>	tick box	tick	
<i>if <b>ticked</b>, as each lot number is labelled, the <b>Next lot number</b> is incremented by the <b>Lot increment</b></i>			
<i>If <b>not ticked</b>, the <b>Next lot number</b> field is not incremented.</i>			
<b>Lot increment</b>	number box	1	
<i>the amount to increment the <b>Next lot number</b> by.</i>			
<b>Selected lot area</b>	real box		
<i>when a string that is not already a lot is selected, the area of the string is calculated and written to this field. If a different area is required, change the value in this field and click <b>Update</b>.</i>			
<b>Pick to Create or Edit</b>	button		
<i>click on the button and then pick a string.</i>			
<i>If the picked string already has Lot attributes set by this option (or other options) then those values will be displayed in the <b>Lot Edit</b> panel, and modified and saved by clicking <b>Update</b> on that panel.</i>			



*If the picked string has not yet had Lot attributes set by this option then the string will be given Lot attributes using the **Plan number**, **Next lot number** and **Selected lot area** fields.*

Continue to the next section [27.4.9.2.3 Generate ADAC Road Edge Types](#) or return to [27.4.9.2 Data Prep](#).

### 27.4.9.2.3 Generate ADAC Road Edge Types

#### Position of menu:

File I/O =>ADAC =>User =>Data prep =>Generate ADAC Road Edge types

There is an ADAC Asset *Transport>RoadEdges>RoadEdge* and it has the element *Type* with the choices

Independent Simple Data Type: [transport\\_edge\\_type](#) [Back to Root Element](#)  
 Constrains: [Feature\\_Transport\\_RoadEdge\\_Type](#)

Type of Road Edge - As per DMR drawing 1033

Restriction of: [\[String\\_32\]](#)

Enumeration:	B1	Barrier Kerb Type 5
Enumeration:	B2	Barrier Kerb with Channel Type 6
Enumeration:	B3	Barrier Kerb with Channel Type 7
Enumeration:	B4	Barrier Kerb with Tray Type 23
Enumeration:	B5	Barrier Kerb with Tray Type 24
Enumeration:	SM1	Semi-Mountable Kerb Type 8
Enumeration:	SM2	Semi-Mountable Kerb Type 10
Enumeration:	SM3	Semi-Mountable Kerb Type 12
Enumeration:	SM4	Semi-Mountable Kerb with Channel Type 14
Enumeration:	SM5	Semi-Mountable Kerb with Channel Type 15
Enumeration:	M1	Mountable Kerb and Channel
Enumeration:	M2	Mountable Kerb and Channel
Enumeration:	M3	Mountable Kerb and Channel
Enumeration:	M4	Mountable Kerb
Enumeration:	M5	Mountable Kerb
Enumeration:	M6	Mountable Kerb
Enumeration:	ER1	Edge Restraint
Enumeration:	ER2	Edge Restraint
Enumeration:	ER3	Edge Restraint
Enumeration:	ER4	Edge Restraint
Enumeration:	ER5	Edge Restraint
Enumeration:	INV600	Concrete Channel Type 22
Enumeration:	INV900	Concrete Channel Type 28
Enumeration:	Bitumen	Bitumen Road Edge treatment
Enumeration:	Concrete	Concrete Road Edge treatment

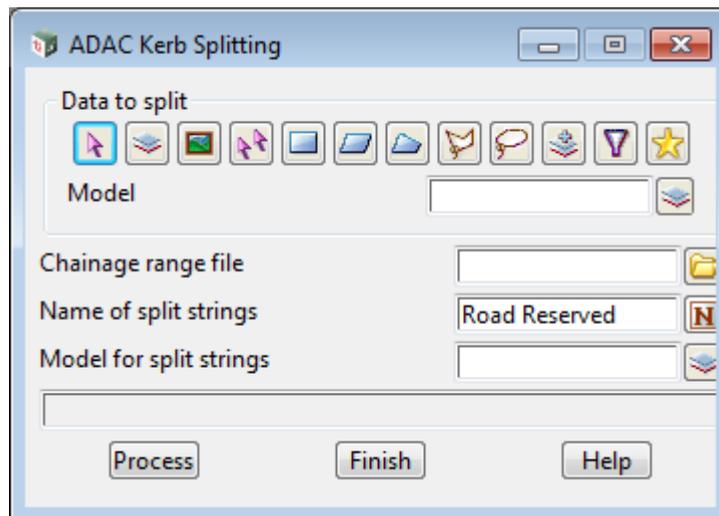
However, many companies do not have a naming convention that differentiates between the choices and may use just the one string name for all of them. So there is no way to automatically know the **Type** which is **mandatory** for an ADAC *Transport>RoadEdges>RoadEdge* Asset.

To help get over this problem, this option takes a file which contains any number of string names and for each string, a list of chainage ranges and kerb types.

For each string listed in this file, the option creates new strings for each of the chainage ranges. The new strings also have a User string attribute called **RoadEdge\_Type** created, and it is given the value of **Type**.

It is the strings produced by this 12dPL, and not the original strings, that ADAC requires, and they are the ones added to the **Design to map to ADAC** or view **Survey to map to ADAC** for processing.

Selecting **Generate ADAC Road Edge types** brings up the **ADAC Kerb Splitting** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Data source type</b>		Model	
-------------------------	--	-------	--

*data selection type - for a full description go to [4.19.3 Data Source](#).*

<b>Data source</b>	input		
--------------------	-------	--	--

*source of strings that is searched for strings to split.*

<b>Chainage range file</b>	file box		
----------------------------	----------	--	--

*the file giving the names of the strings to be split and the chainage ranges to split the string into and the Type that goes with that chainage range. Each new string is given a text string attribute called RoadEdge\_Type which has the given kerb\_type\_i.*

*In the file, the format for giving the chainage ranges and kerb types for a string is:*

```
STRING model_name ->string_name
  start_chainage_1 end_chainage_1 kerb_type_1
  start_chainage_2 end_chainage_2 kerb_type_2
  start_chainage_i end_chainage_i kerb_type_i
```

*For example,*

```
STRING MC01 DESIGN->KIR
0 100 M1
100 9999 B1
```

<b>Name of split strings</b>	text box		
------------------------------	----------	--	--

*if **blank**, the original string name is used for the new strings.*

*If **not blank**, the created split strings will be given this name.*

<b>Model for split strings</b>	model box	available models	
--------------------------------	-----------	------------------	--

*the model for the created split strings.*

<b>Process</b>	button		
----------------	--------	--	--

*go and create the split strings.*

Continue to the next section [27.4.9.2.4 Create Points from Other Data](#) or return to [27.4.9.2 Data Prep](#).

## 27.4.9.2.4 Create Points from Other Data

### Position of menu:

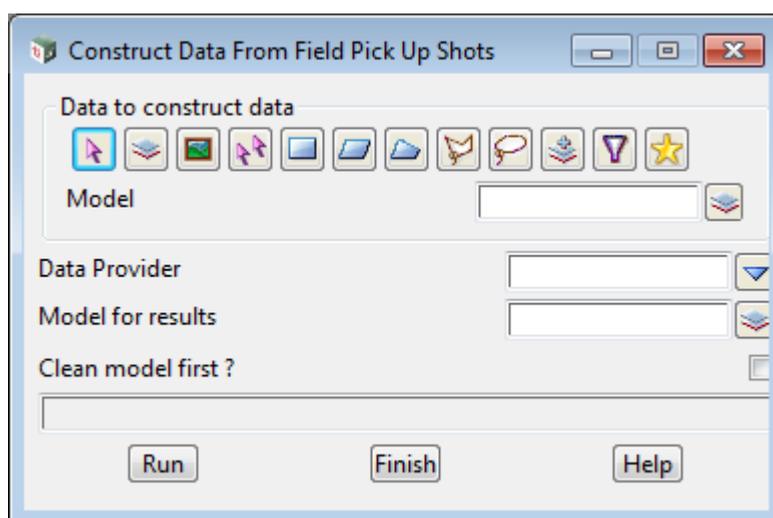
File I/O =>ADAC =>User =>Data prep =>Create points from other data

Sometimes strings are not suitable to go straight to ADAC but can be used to create suitable strings without the user having to do anything manually.

For example, ADAC only has a one point object for signs and a user picks up large signs as a two vertex string. This option takes the two vertex large sign and creates a new one vertex string that **can** be mapped to ADAC as an ADAC sign.

This option has various methods of creating ADAC suitable data.

Selecting **Create points from other data** brings up the **Construct Data from Field Pick Up Shots** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of strings that is process to generate strings suitable for ADAC.</i>			
<b>Data Provider</b>	choice box		available data providers
<i>selects the table of modifications to make.</i>			
<b>Model for results</b>	model box		available models
<i>model that the processed strings are added to.</i>			
<b>Clean model first ?</b>	tick box		
<i>if <b>ticked</b>, the model <b>Model for results</b> is cleaned before any strings are added to it.</i>			
<i>If <b>not ticked</b>, the model <b>Model for results</b> is <b>NOT</b> cleaned.</i>			
<b>Run</b>	button		
<i>process the data.</i>			

Continue to the next section [27.4.9.2.5 Create Pipes from Points](#) or return to [27.4.9.2 Data Prep](#)

### 27.4.9.2.5 Create Pipes from Points

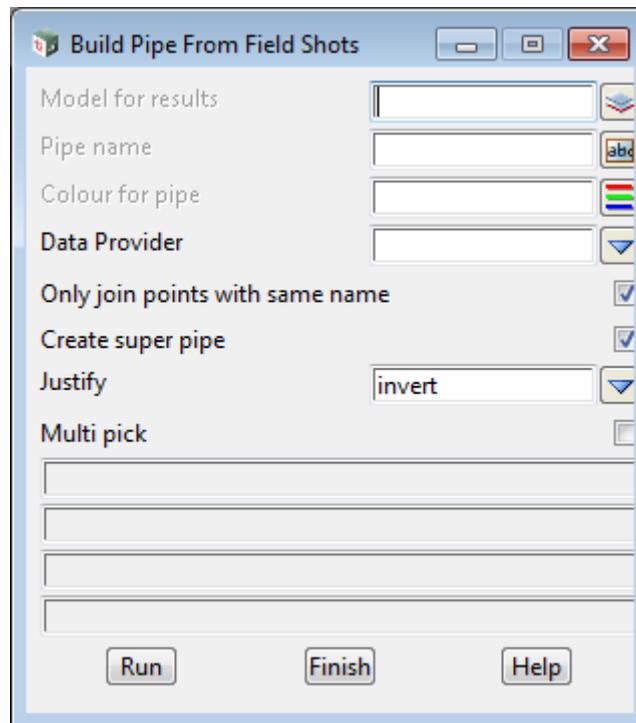
#### Position of menu:

File I/O =>ADAC =>User =>Data prep =>Create pipes from points

Sometimes pipes can not be picked up as string in the field and only one vertex strings are picked up at say the ends of the pipes.

This option joins two selected one vertex strings to create a two vertex pipe string.

Selecting **Create pipes from points** brings up the **Build Pipe from Field Shots** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Model for results</b> <i>model that the create pipes are added to.</i>	model box		available models
<b>Pipe name</b> <i>if <b>not blank</b>, the created string is given this name. If blank,</i>	text box		
<b>Colour for pipe</b> <i>if <b>not blank</b>, the created string is given this colour. If blank,</i>	colour box		available colours
<b>Data Provider</b> <i>selects the table of attributes to use for diameter etc.</i>	choice box		available data providers
<b>Only join points with same name</b> <i>if <b>ticked</b>, after picking the first one vertex string, the second one vertex string must have the same name as the first picked string.</i>	tick box		
<b>Create super pipe</b> <i>if <b>ticked</b>, and there is an attribute specified in the selected <b>Data provider</b> for diameter, then a super string pipe with the given diameter is created.</i>	tick box		

- Justify** choice box invert, centre, obvert  
*justification to use if a super pipe is created.*
- Multi pick** tick box  
*if **ticked**, once one pipe is created, the option starts asking for the first point of a new pipe.*
- Run** button  
*start the option.*

Continue to the next section [27.4.9.2.6 Extend Attributes](#) or return to [27.4.9.2 Data Prep](#).

## 27.4.9.2.6 Extend Attributes

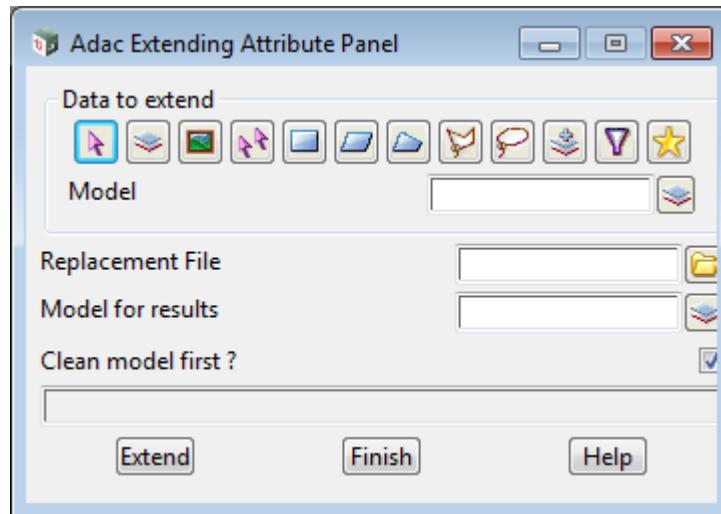
### Position of menu:

File I/O =>ADAC =>User =>Data prep =>Extend Attributes

This option replaces text attribute values with different attribute values.

It can also work on attribute names.

Selecting Extend Attributes brings up the **ADAC Extending Attribute** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

<b>Data source type</b>		Model	
-------------------------	--	-------	--

*data selection type - for a full description go to [4.19.3 Data Source](#).*

<b>Data source</b>	input		
--------------------	-------	--	--

*source of data to process.*

<b>Replacement File</b>	file box		available files
-------------------------	----------	--	-----------------

*The file is three words per line, and each word must be enclosed in quotation marks - “.*

*“first\_word\_attribute\_path\_name” ”second\_word\_value” “third\_word\_new\_word”*

*For example*

*“Material” “CONC” “CONCRETE”*

*The first word is the full attribute path name, the second word is the value of the attribute itself, and the third word is the what the value gets converted to.*

*String, vertex and segment attributes will be looked at for the attribute path name.*

*For example*

*“Material” “CONC” “CONCRETE”*

*will look in string, vertex and segment attributes to find the attribute path named “Material”, and if its value is “CONC”, this will be converted to “CONCRETE”*

*“Type/VEHICLE” “CONVOY” “CONVOYAHA”*

*will look in string, vertex and segment attributes and find attribute path named “Type/VEHICLE” and if its value is “CONVOY”, this will be converted to “CONVOYAHA”*

**Special Case:**

If the first word is left blank, the option looks for the second word inside any attribute path name and any attribute value, and replaces it with the third word in the attribute path name and/or the attribute value.

For example,

"" "PRO" "PROFESSIONAL"

Will look in string, vertex and segment attributes and find attribute names with the word PRO in them and any attribute values with the word PRO in them, and changes the PRO to "PROFESSIONAL"

**Note**

The quotation marks are compulsory because 12d allows spaces for attribute names and values. Two quotation marks next to each other indicates an empty attribute path.

**Model for results**                      model box    available models

*all the selected data is processed and copies (modified or not) are placed in this model.*

**Clean model first?**                      tick box                                      ticked

*if ticked, the model **Model for results** is cleaned before any data is added to it.*

**Extend**                                      button

*runs the option.*

Continue to the next section [27.4.9.2.7 Set Stormwater/Sewer Property](#) or return to [27.4.9.2 Data Prep](#).

### 27.4.9.2.7 Set Stormwater/Sewer Property

#### Position of menu:

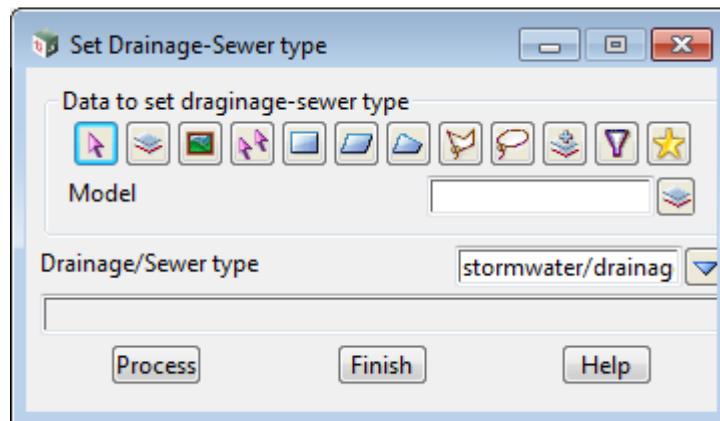
File I/O =>ADAC =>User =>Data prep =>Set stormwater/sewer purpose

Up to **12d Model 10**, there was no way of easily differentiating between a drainage or sewer string. Users normally kept them in different models.

In **12d Model 11**, there is a string property (called **Purpose** on the **Create Drainage String** menu) which is set to either **stormwater/drainage** or **wastewater/sewer**. So the string itself now knows if it is a drainage or a sewer string.

This option sets the **Purpose** for drainage/sewer strings that have come from **12d Model 10** and so do not have a **Purpose** already set, or may have it defaulted to **stormwater/drainage**.

Selecting Set stormwater/sewer purpose brings up the **Set Drainage/Sewer Purpose** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Data source type</b>		Model	
<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>			
<b>Data source</b>	input		
<i>source of strings to process.</i>			
<b>Drainage/Sewer purpose</b>	choice box	stormwater/drainage, wastewater/sewer	
<i>the Purpose property of the selected strings are set to this value.</i>			
<b>Run</b>	button		
<i>set the Purpose property of the selected strings.</i>			

Continue to the next section [27.4.9.2.8 Set a Drainage-Sewer Pit and Pipe Type Attribute](#) or return to [27.4.9.2 Data Prep](#).

### 27.4.9.2.8 Set a Drainage-Sewer Pit and Pipe Type Attribute

#### Position of menu:

File I/O =>ADAC =>User =>Data prep =>Set pit type, pipe type attribute

When drainage and sewer strings are created, the vertices can have a **Pit Type** and the segments a **Pipe Type**.

When the Drainage Network Editor (DNE) is used, for each vertex, a vertex attribute called **pit\_type** is created with the vertex value of **Pit Type**. Similarly for each segment, a segment attribute called **pipe\_type** is created with the segment value of **Pipe Type**.

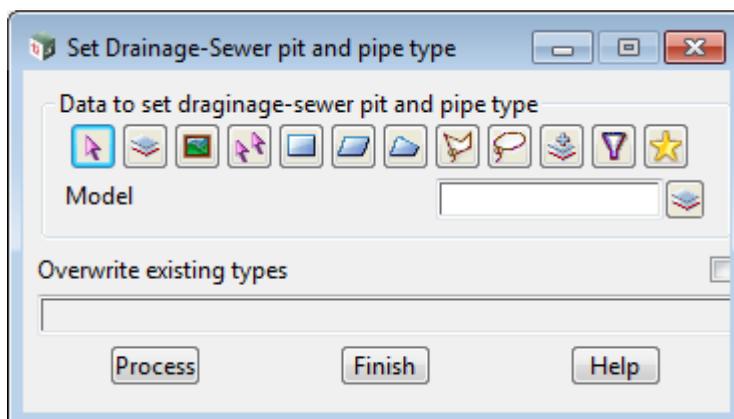
So after using the DNE, the attributes **pit\_type** and **pipe\_type** are available to be used in the ADAC **Map File**. However if the DNE is not run, the attributes **pit\_type** and **pipe\_type** will not exist.

This option does the same thing as the DNE for creating the **pit\_type** and **pipe\_type** attributes.

That is, for each vertex, a vertex attribute called **pit\_type** is created with the vertex value of **Pit Type** and for each segment, a segment attribute called **pipe\_type** is created with the segment value of **Pipe Type**.

So in case th DNE has not been run, this option creates the attributes **pit\_type** and **pipe\_type** so they can be used in the **Map File**.

Selecting Set pit type, pipe type attribute brings up the **Set Attribute for Drainage/Sewer Pit and Pipe Types** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
-------------------	------	----------	--------

#### Data source type

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

#### Data source

input

*source of strings to process.*

#### Overwrite existing type

tick box

*if **ticked** and a **pit\_type/pipe\_type** already exists for a vertex/segment, then it is over written by the value of **Pit type/Pipe Type** for the vertex/segment.*

*if **not ticked** then if a **pit\_type/pipe\_type** already exists for a vertex/segment, then it is **not** modified.*

#### Run

button

*set the **pit\_type** and **pipe\_type**'s of the selected strings vertices and segments.*

Continue to the next section [27.4.9.2.9 Move/Link Pipe Ends to Pit Centres](#) or return to [27.4.9.2 Data Prep](#).

### 27.4.9.2.9 Move/Link Pipe Ends to Pit Centres

#### Position of menu:

File I/O =>ADAC =>User =>Data prep =>Move/link pipe ends to pit centres

For some Authorities, the ends of the strings representing the Sewerage PipesNonPressure, Sewerage PipesPressure, and Stormwater Pipe are not to be where they are picked up going into a maintenance hole or pit, but are to end on the point representing the maintenance hole or pit.

strings are created, the vertices can have a **Pit Type** and the segments a **Pipe Type**.

When the Drainage Network Editor (DNE) is used, for each vertex, a vertex attribute called **pit\_type** is created with the vertex value of **Pit Type**. Similarly for each segment, a segment attribute called **pipe\_type** is created with the segment value of **Pipe Type**.

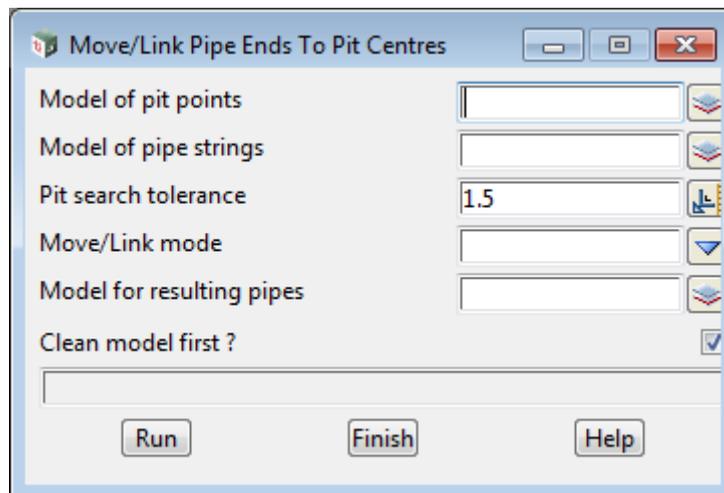
So after using the DNE, the attributes pit\_type and pipe\_type are available to be used in the ADAC **Map File**. However if the DNE is not run, the attributes pit\_type and pipe\_type will not exist.

This option does the same thing as the DNE for creating the **pit\_type** and **pipe\_type** attributes.

That is, for each vertex, a vertex attribute called **pit\_type** is created with the vertex value of **Pit Type** and for each segment, a segment attribute called **pipe\_type** is created with the segment value of **Pipe Type**.

So in case th DNE has not been run, this option creates the attributes **pit\_type** and **pipe\_type** so they can be used in the **Map File**.

Selecting **Move/link pipe ends to pit centres** brings up the **Move/Link Pipe Ends to Pit Centres** panel



The fields and buttons used in this panel have the following functions.

Field Description	Type	Defaults	Pop-Up
<b>Model of pit points</b> <i>the points representing the maintenance holes or pits.</i>			available models
<b>Model of pipe strings</b> <i>the strings representing the pipes that go between the maintenance holes or pits.</i>			available models
<b>Model for resulting pipes</b> <i>copies of the pipe strings are modified and the resulting pipes strings are added to the model.</i>			available models
<b>Pit search tolerance</b> <i>this is the distance to search from each end of a pipe string to find the closest maintenance hole or pit.</i>	real box		

*If nothing is found within this distance then that end of the string is not modified.*

**Move/Link mode**                      choice box                      Move pipe ends to closest pits  
Add link at pipe ends to closest pits

*if **Move pipe ends to closest pits**, then for each end of a pipe strings, if there is a pit point within the **Pit search tolerance** distance then that end is given the (x,y) coordinate of the closest pit point. If no pit point is found then that end of the pipe string is not moved.*

*if **Add link at pipe ends to closest pits**, then for each end of a pipe strings, if there is a pit point within the **Pit search tolerance** distance then an extra segment is added to that end of the pipe string going from the existing end of the pipe string to the closest pit point. If no pit point is found then no extra segment is added to that end of the pipe string.*

**Clean model first?**                      tick box

*if **ticked**, the **Model for resulting pipes** is cleaned before processing begins.*

*if **not ticked** then the **Model for resulting pipes** is not cleaned.*

**Run**    button

*set the pit\_type and pipe\_type's of the selected strings vertices and segments.*

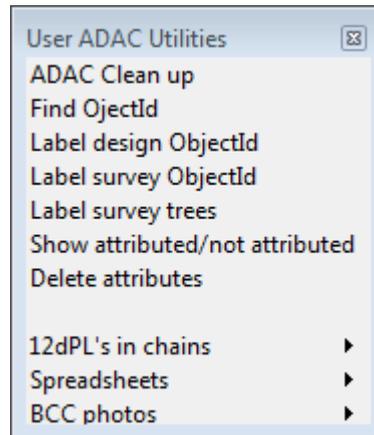
Return to [27.4.9.2 Data Prep](#) or [27.4.9 User ADAC](#).

## 27.4.9.3 User ADAC Utilities

**Position of menu:** File I/O =>ADAC =>User =>Utilities

A new project to be set up for producing ADAC XML needs to be set up a certain way for the ADAC Chains to work.

There is a different setup for a **Survey** Project than for a **Design** Project.



See

[27.4.9.3.1 ADAC Clean Up](#)

[27.4.9.3.2 Find Object Id](#)

[27.4.9.3.3 Label Design/Survey Object Id](#)

[27.4.9.3.4 Show Attributed/Not Attributed](#)

[27.4.9.3.5 Delete ADAC Attributes](#)

[27.4.9.3.6 12dPL's in Chains](#)

### 27.4.9.3.1 ADAC Clean Up

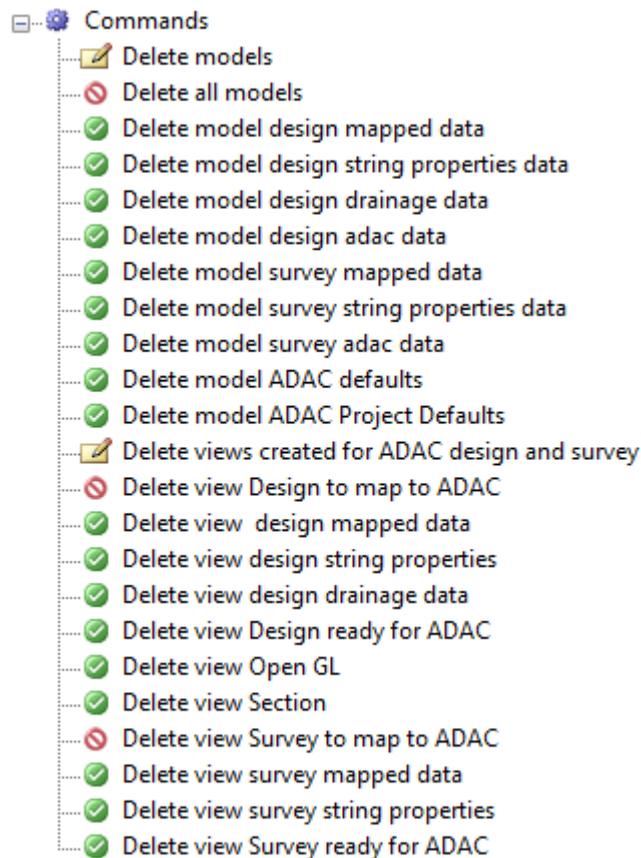
**Position of option on menu:** File I/O =>ADAC =>User =>Utilities =>ADAC clean up

This option runs a chain that deletes all the models created by the **Setting up for survey** and **Setting up for design** options.

The chain also deletes all the views, except for the views *Design to map to ADAC* and *Survey to map to ADAC*, created by the **Setting up for survey** and **Setting up for design** options.

Selecting **ADAC clean up** runs the chain

```
$LIB/ADAC_clean_up.chain
```



The chain needs no interaction with the user; it just runs and ends.

Continue to the next section [27.4.9.3.2 Find Object Id](#) or return to [27.4.9.3 User ADAC Utilities](#).

### 27.4.9.3.2 Find Object Id

**Position of option on menu:** File I/O =>ADAC =>User =>Utilities =>Find Objectid

From the ADAC XML Schema, the **Objectid**

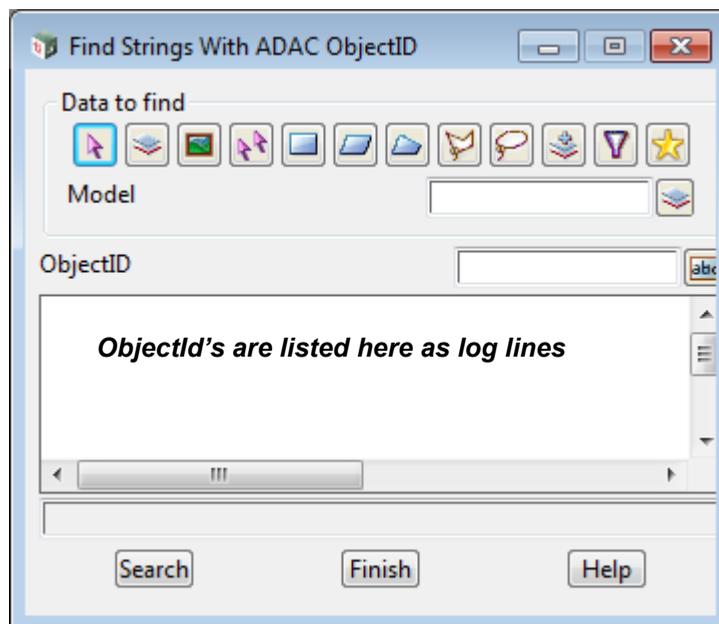
*Represents a place for an object identifier, usually generated by the data creation software. Also useful as a placeholder for record ID generated by back-end databases or GIS*

*The Objectid is permitted to be nil because the capturing system may not have the capacity to generate one. If features are exported from a GIS or Asset Management system, however, it is highly recommended to carry them out into this element.*

When the ADAC Survey and ADAC Design chains are run, the ADAC Assets are all given a unique ADAC Objectid by **12d Model** and this is given in the ADAC Report (see [27.4.5 Report](#)) and any ADAC XML file that is created (see [27.4.6 Write ADAC XML File](#)).

The **Find Objectid** option lists as log lines the Objectid of all ADAC Assets whose Objectids start with given text. Clicking on an Objectid in the list highlights it in any Plan view that it is on.

Selecting Find Objectid brings up the **Find Strings with ADAC Objectid** panel:



**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to be searched for any with ObjectIDs that start with the characters in **ObjectID start characters**.*

**Objectid start characters** text box

*the characters the ObjectIDs of the strings must start with.*

#### List Area

*the ObjectID of each string that starts with the characters in the **ObjectID start characters** is listed in this area as a log line. Clicking on an Objectid in the list will pan to and highlight the string in any Plan views that the string is on.*

**Search**

button

*search for all ADAC strings that start with the characters in **ObjectID start characters**.*

Continue to the next section [27.4.9.3.3 Label Design/Survey Object Id](#) or return to [27.4.9.3 User ADAC Utilities](#).

### 27.4.9.3.3 Label Design/Survey Object Id

[check why it is not working]

**Position of option on menu:** File I/O =>ADAC =>User =>Utilities =>Label design Objectid

**Position of option on menu:** File I/O =>ADAC =>User =>Utilities =>Label survey Objectid

From the *ADAC XML Schema*, the **Objectid**

*Represents a place for an object identifier, usually generated by the data creation software. Also useful as a placeholder for record ID generated by back-end databases or GIS*

*The Objectid is permitted to be nil because the capturing system may not have the capacity to generate one. If features are exported from a GIS or Asset Management system, however, it is highly recommended to carry them out into this element.*

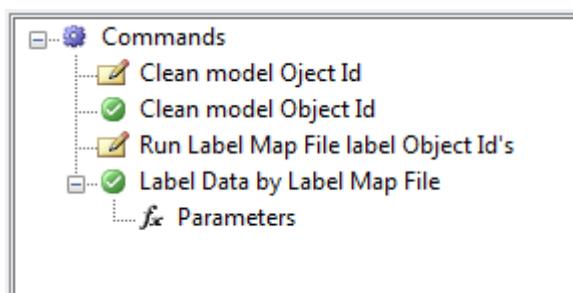
This option runs an Interactive chain that includes a **Label Data by Label Map File** panel that creates labels for all the ADAC assets in the model **design adac data** or **survey adac data** and cleans and adds them to the model **Objectid**.

**Note:** the model **design adac data/survey adac data** is the model that the *ADAC Design/Survey* chain puts all the ADAC Assets in.

Selecting **Label design/surveyObjectid** runs the Interactive chain

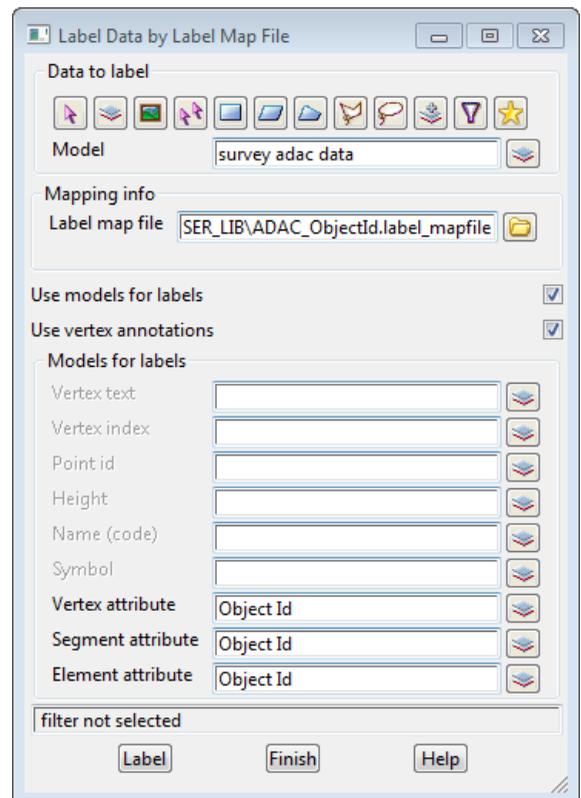
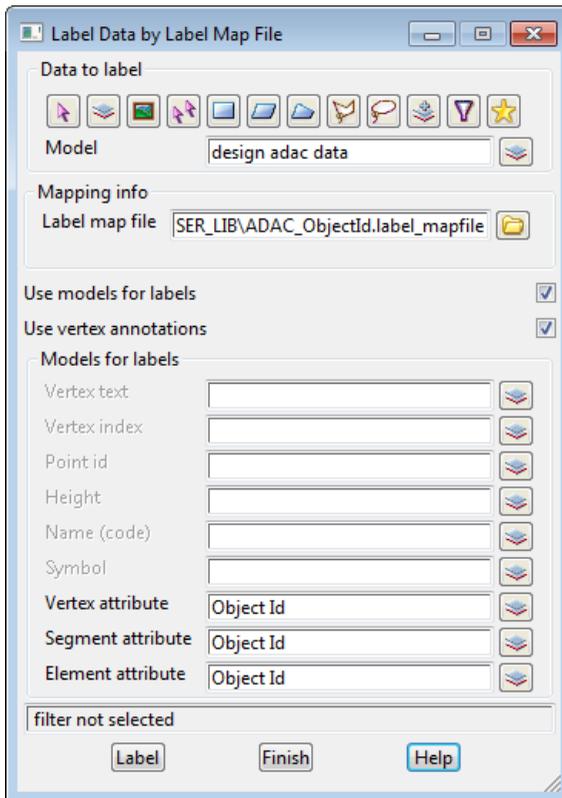
\$LIB/ADAC\_Label\_design\_Objectid.chain

\$LIB/ADAC\_Label\_survey\_Objectid.chain



The chain first cleans out the model **Objectid**

It then runs a **Label Data by Label Map File** panel in Interactive mode so the panel is placed on the screen.



Click on **Label** and when the labelling is finished, click on **Finish**.

The chain then ends.

Continue to the next section [27.4.9.3.4 Show Attributed/Not Attributed](#) or return to [27.4.9.3 User ADAC Utilities](#).

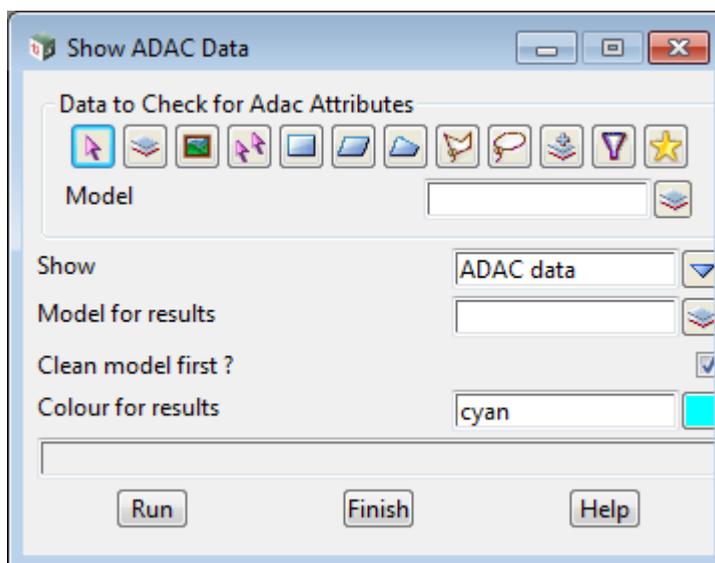
### 27.4.9.3.4 Show Attributed/Not Attributed

**Position of option on menu:** File I/O =>ADAC =>User =>Utilities =>Show attributed/not attributed

When checking whether your 12d Map File is doing the correct thing, it is often necessary to know which strings have been given an ADAC attribute group and hence represent an ADAC Asset, and which ones weren't.

This option searches for all strings that are either ADAC Assets, or are not ADAC Assets, and make copies of the string and adds them to a given model.

Selecting **Show attributes/not attributes** brings up the **Show ADAC Attributed Data** panel



<b>Data source type</b>		Model	
	<i>data selection type - for a full description go to <a href="#">4.19.3 Data Source</a>.</i>		
<b>Data source</b>		input	
	<i>source of data to be check if it has/doesn't have ADAC attributes.</i>		
<b>Show</b>	choice box	ADAC data	ADAC data, non ADAC data
	<i>if <b>ADAC data</b>, for each string in the data source that is an ADAC Asset (i.e. has the ADAC attributes), a copy of the string is made in the colour given in <b>Colour for results</b> and is added to the model in <b>Model for results</b>.</i>		
	<i>if <b>non ADAC data</b>, for each string in the data source that is NOT an ADAC Asset (i.e. doesn't have the ADAC attributes), a copy of the string is made in the colour given in <b>Colour for results</b> and is added to the model in <b>Model for results</b>.</i>		
<b>Model for results</b>	model box		available models
	<i>the model for the created strings.</i>		
<b>Clean model first</b>	tick box		
	<i>if <b>ticked</b>, the model <b>Model for results</b> is cleaned before any strings are added to it.</i>		
	<i>If <b>not ticked</b>, the model <b>Model for results</b> is NOT cleaned.</i>		
<b>Colour for results</b>	colour box	cyan	available colours
	<i>the colour to give the created strings.</i>		
<b>Run</b>	button		
	<i>finds all the ADAC/not ADAC strings.</i>		

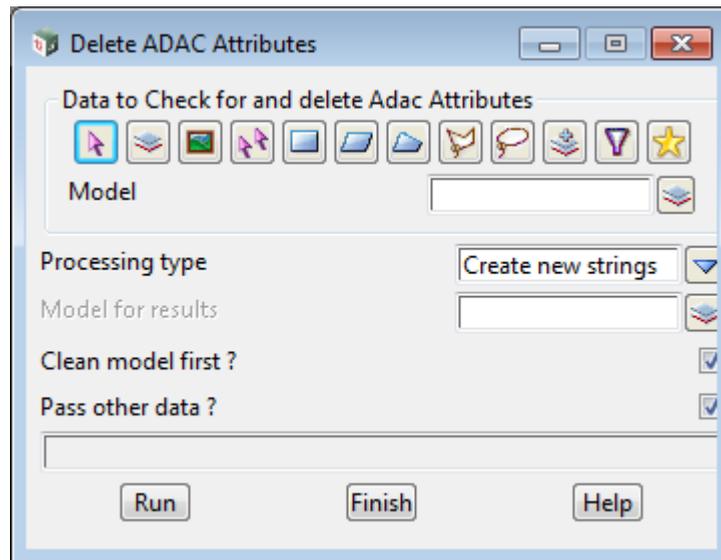
Continue to [27.4.9.3.5 Delete ADAC Attributes](#) or return to [27.4.9.3 User ADAC Utilities](#).

### 27.4.9.3.5 Delete ADAC Attributes

**Position of option on menu:** File I/O =>ADAC =>User =>Utilities =>Delete attributes

This option removes the ADAC attributes from strings (and only those attributes).

Selecting **Delete attributes** brings up the **Delete ADAC Attributes** panel:



**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to process for deleting ADAC attributes.*

**Processing type**

choice box

Create new strings

Use existing strings  
Create new strings

*if **Create new strings**, for each string in the data source that is an ADAC Asset (i.e. has the ADAC attributes), a copy of the string is made and the ADAC attributes deleted from the **copy** of the string and added to the model **Model for results**.*

*if **Use existing strings**, for each string in the data source that is an ADAC Asset (i.e. has the ADAC attributes), the ADAC attributes are deleted from the string.*

**Model for results**

model box

available models

*when **Processing type** is **Create New Strings** this is the model for the created strings.*

**Clean model first ?**

tick box

*if **ticked**, the model **Model for results** is cleaned before any strings are added to it.  
If **not ticked**, the model **Model for results** is **NOT** cleaned.*

**Pass other data ?**

tick box

*if **ticked** and **Processing type** is **Create new strings**, then any strings that are not ADAC Assets are also copied and added to the model **Model for results**.  
Otherwise nothing is done with strings that are not ADAC Assets.*

**Run**

button

*removed the ADAC attributes from the selected strings.*

Continue to [27.4.9.3.6 12dPL's in Chains](#) or return to [27.4.9.3 User ADAC Utilities](#).

### 27.4.9.3.6 12dPL's in Chains

**Position of menu:** File I/O =>ADAC =>User =>Utilities



There are the options used in the ADAC Design and Survey chains.

See

[27.4.9.3.6.1 Separate Assigned/Unassigned ADAC Data](#)

[27.4.9.3.6.2 Set ADAC Attributes from String Properties](#)

[27.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data](#)

[27.4.9.3.6.4 Set ADAC Attributes from User Attributes](#)

[27.4.9.3.6.5 Set ADAC ObjectID from String UID](#)

[27.4.9.3.6.5 Set ADAC ObjectID from String UID](#)

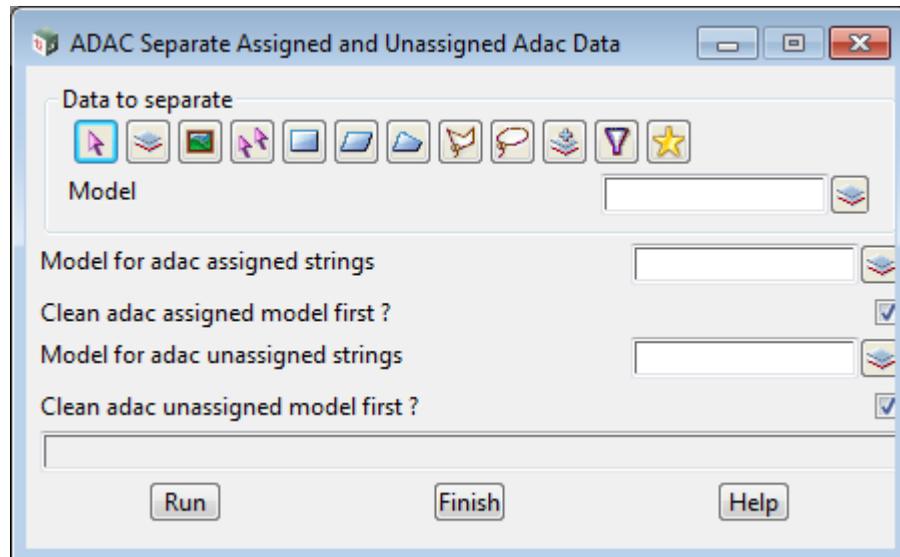
### 27.4.9.3.6.1 Separate Assigned/Unassigned ADAC Data

#### Position of option on menu:

File I/O =>ADAC =>User =>Utilities =>12dPLs in chains =>Separate assigned data

This option looks at the data in the given Data Source and copies the strings that have been assigned (marked) as ADAC Assets to one model and all the data that has not yet been assigned to another model.

Selecting **Separate Assigned data** brings up the **ADAC Separate Assigned and Unassigned Data** panel.



#### Data source type

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

#### Data source

input

*source of data to process.*

#### Model for ADAC assigned strings

model box

available models

*any strings that have been assigned (marked) as ADAC assets are copied to this model.*

#### Clean ADAC assigned model first ?

tick box

*if ticked, the model **Model for ADAC assigned strings** is cleaned before any strings are added to it.  
If not ticked, the model **Model for ADAC assigned strings** is NOT cleaned.*

#### Model for ADAC unassigned strings

model box

available models

*any strings that have NOT been assigned (marked) as ADAC assets are copied to this model.*

#### Clean ADAC unassigned model first ?

tick box

*if ticked, the model **Model for ADAC unassigned strings** is cleaned before any strings are added to it.  
If not ticked, the model **Model for ADAC assigned strings** is NOT cleaned.*

#### Run

button

*separate the strings that have been assigned/unassigned as ADAC Assets into separate models.*

*The copied strings are added to the model **Model for results**.*

Continue to the next section [27.4.9.3.6.2 Set ADAC Attributes from String Properties](#) or return to [27.4.9.3.6 12dPL's in Chains](#)

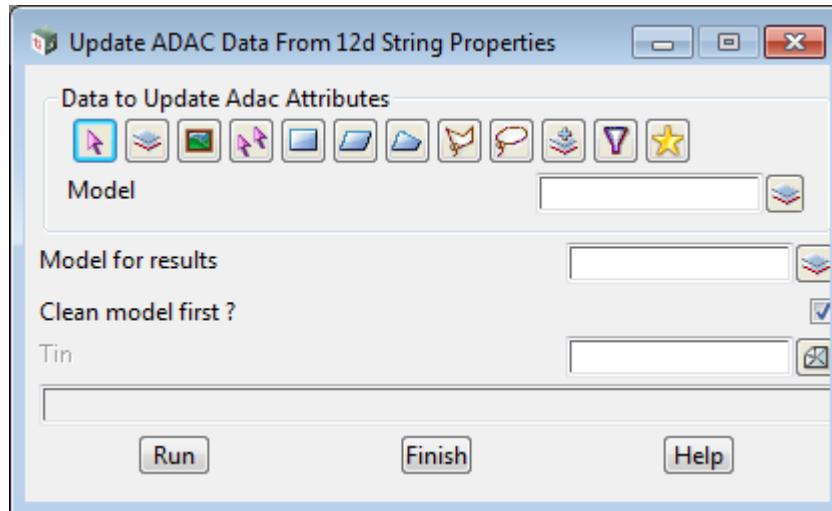
### 27.4.9.3.6.2 Set ADAC Attributes from String Properties

#### Position of option on menu:

File I/O =>ADAC =>User =>Utilities =>12dPLs in chains =>Set ADAC attributes from string properties

Many of the ADAC entities can be calculated directly from the 12d strings and some z values may be taken from a tin.

Selecting **Set ADAC attributes from 12d properties** brings up the **Update ADAC Data from 12d String Properties** panel.



#### Data source type

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

#### Data source

input

*source of data to process.*

#### Model for results

model box

available models

*model that the processed strings are added to.*

#### Clean model first ?

tick box

*if **ticked**, the model **Model for results** is cleaned before any strings are added to it. If **not ticked**, the model **Model for results** is **NOT** cleaned.*

#### Tin

tin box

available tins

*tin that can be used to get z-values from.*

#### Run

button

*update the ADAC Asset attributes with values from the 12d string's properties.*

*If the string has been marked as an ADAC Asset then it is copied and then for the copied string, any relevant ADAC elements updated from the properties of the 12d string and the tin.*

*Strings that are not marked as ADAC Assets are **NOT** copied*

*The copied strings are added to the model **Model for results**.*

## ADAC Attributes Generated from 12d String Geometry

	ADAC path	Action	Feature name	Nullability	T
1.1.0	Adac/Contour/Elevation_m	z value from string	Surface	No	f
1.1.0	Adac/SpotHeight/Elevation_m	z value from vertex	Surface	No	f
1.1.0	Adac/RoadEdge/Length_m	calculate 3d length of string	Transport	Yes	f
1.1.0	Adac/SubsoilDrain/Length_m	calculate 3d length of string	Transport	Yes	f
1.1.0	Adac/RoadIsland/Area_sqm	calculate area of string	Transport	Yes	f
1.1.0	Adac/Parking/Surface/SurfaceArea_sqm	calculate plan area of a string	Transport	Yes	f
1.1.0	Adac/BarrierContinuous/Length_m	calculate 3d length of string	OpenSpace	No	f
	Adac/RetainingWall/Length	calculate 3d length of string	OpenSpace	No	f
	Adac/ElectricalConduit/Size	take diameter from super string	OpenSpace	No	in
	Adac/ElectricalConduit/Diameter_mm	take diameter from super string	OpenSpace	No	in
	Adac/ElectricalConduit/Length_m	calculate 3d length of string	OpenSpace	No	f
1.1.0	Adac/Pipe/Diameter_mm	take diameter from super string	WaterSupply	No	in
1.1.0	Adac/Pipe/Length_m	calculate 3d length of string	WaterSupply	Yes	f
1.1.0	Adac/Maintenancehole/Surfacelevel_m	z value from string	WaterSupply	No	f
	Adac/Manhole/Surfacelevel_m	z value from string	Sewerage	No	f
	Adac/Maintenancehole/Surfacelevel_m	z value from string	Sewerage	No	f
	Adac/Connection/HCB_Length	calculate 3d length of string	Sewerage	No	f
	Adac/Connection/Length_m	calculate 3d length of string	Sewerage	No	f
1.1.0	Adac/Connection/Invertlevel_m	calculate us invert level (higher end)	Sewerage	No	f
1.1.0	Adac/Connection/Surfacelevel_m	calculate tin us surface level (higher end, tin required)	Sewerage	No	f
1.1.0	Adac/PipeNonPressure/DS_Surfacelevel_m	calculate tin ds surface level (higher end, tin required)	Sewerage	No	f
1.1.0	Adac/PipeNonPressure/AverageDepth_m	get average depth based on other attributes	Sewerage	No	f
1.1.0	Adac/PipeNonPressure/PipeGrade	calculate grade	Sewerage	Yes	f
1.1.0	Adac/PipeNonPressure/Length_m	calculate 3d length	Sewerage	Yes	f
1.1.0	Adac/PipeNonPressure/US_Invertlevel_m	calculate us invert level (higher end)	Sewerage	no	f
1.1.0	Adac/PipeNonPressure/DS_Invertlevel_m	calculate ds invert level (lower end)	Sewerage	no	f
1.1.0	Adac/PipeNonPressure/Diameter_mm	calculate diameter	Sewerage	no	f
1.1.0	Adac/Pipe/US_Surfacelevel_m	calculate tin us surface level (higher end, tin required)	StormWater	No	f
1.1.0	Adac/Pipe/DS_Surfacelevel_m	calculate tin ds surface level (higher end, tin required)	StormWater	No	f
1.1.0	Adac/Pit/Depth_m	calculate stormwater pit depth from other attributes	StormWater	No	f

4.0.0 & 4.1.0	Adac/Pipe/Length_m	calculate 3d length	StormWat
4.0.0 & 4.1.0	Adac/Pipe/Grade	calculate grade	StormWat
4.0.0 & 4.1.0	Adac/SurfaceDrain/Length_m	calculate 3d length	StormWat
4.0.0 & 4.1.0	Adac/Pipe/US_Invertlevel_m	calculate us invert level (higher end)	StormWat
4.0.0 & 4.1.0	Adac/Pipe/DS_Invertlevel_m	calculate ds invert level (lower end)	StormWat
4.0.0 & 4.1.0	Adac/Pipe/DS_Invertlevel_m	calculate circ diameter	StormWat
4.0.0 & 4.1.0	Adac/Pipe/PipeStructure/CircPipe/Diameter_mm	calculate box height	StormWat
4.0.0 & 4.1.0	Adac/Pipe/PipeStructure/BoxPipe/Height_mm	calculate box width	StormWat
4.0.0 & 4.1.0	Adac/Pipe/PipeStructure/BoxPipe/Width_mm	calculate 3d length of string	StormWat
4.0.0 & 4.1.0	Adac/SurfaceDrain/Length_m		
4.0.0 & 4.1.0	Adac/Annotation/Text	get the actual text of the text string	Enhanceme
4.0.0 & 4.1.0	Adac/Annotation/Justification	get justification of text string	Enhanceme
4.0.0 & 4.1.0	Adac/Annotation/Type	get text type of text string	Enhanceme
4.0.0 & 4.1.0	Adac/Annotation/Font	get text font of text string	Enhanceme
4.0.0 & 4.1.0	Adac/Annotation/Height_m	get text height of text string	Enhanceme
4.0.0 & 4.1.0	Adac/Annotation/Width_m	get text width of text string	Enhanceme
4.0.0 & 4.1.0	Adac/Annotation/Rotation	get text rotation of text string	Enhanceme

In the *ADAC Survey* and *Design* chains, this option is called and passed arguments for all the panel field and so runs without needing to display a panel.

See [27.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey](#) and [27.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design](#).

Continue to the next section [27.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data](#) or return to [27.4.9.3.6 12dPL's in Chains](#).

### 27.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data

#### Position of option on menu:

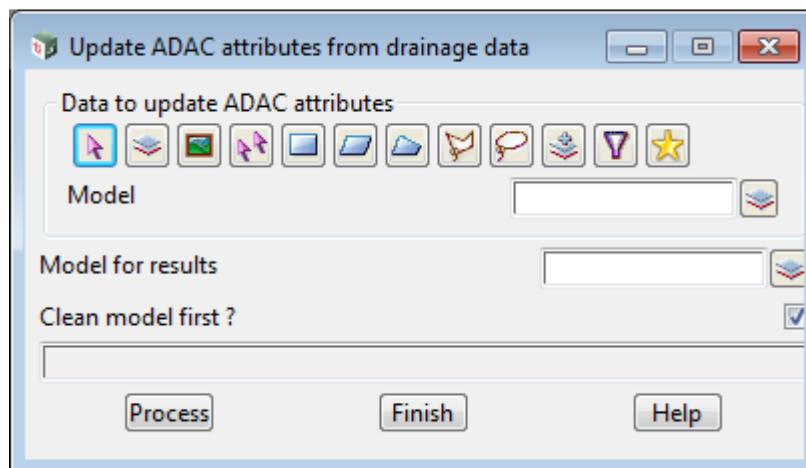
File I/O =>ADAC =>User =>Utilities =>12dPLs in chains =>Set ADAC attributes from drainage/ sewer data

**This section of documentation is a work in progress and will be updated in subsequent releases.**

Drainage and sewer strings have an incredible amount of data that can go directly into the ADAC *StormWater Pits and Pipes* and the *Sewerage MaintenanceHoles* and *NonPressurePipes*.

Many of the ADAC entities can be calculated directly from the 12d strings and some values may be taken from the tins **survey\_finished\_tin** or **design\_finished\_tin**.

Selecting **Set ADAC attributes from Drainage & Sewer Data** brings up the **Update ADAC Data from Drainage Data** panel:



#### Data source type

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

#### Data source

input

*source of data to process.*

#### Model for results

model box

available models

*model that the processed strings are added to.*

#### Clean model first ?

tick box

*if ticked, the model **Model for results** is cleaned before any strings are added to it. If not ticked, the model **Model for results** is NOT cleaned.*

#### Process

button

*update the ADAC Asset attributes with values from the 12d strings properties.*

*If the string has been marked as an ADAC Asset then it is copied and then for the copied string, any relevant ADAC elements updated from the properties of the 12d string and the tin.*

*Strings that are not marked as ADAC Assets are NOT copied*

*The copied strings are added to the model **Model for results**.*

In the *ADAC Design* chain, this option is called and passed arguments for all the panel field and so runs without needing to display a panel.

See [27.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties](#).

Continue to the next section [27.4.9.3.6.5 Set ADAC ObjectID from String UID](#) or return to [27.4.9.3.6 12dPL's in Chains](#).

### 27.4.9.3.6.4 Set ADAC Attributes from User Attributes

NO LONGER USED

**Position of option on menu:**

File I/O =>ADAC =>User =>Utilities =>12dPLs in chains =>Set ADAC attributes from user attributes

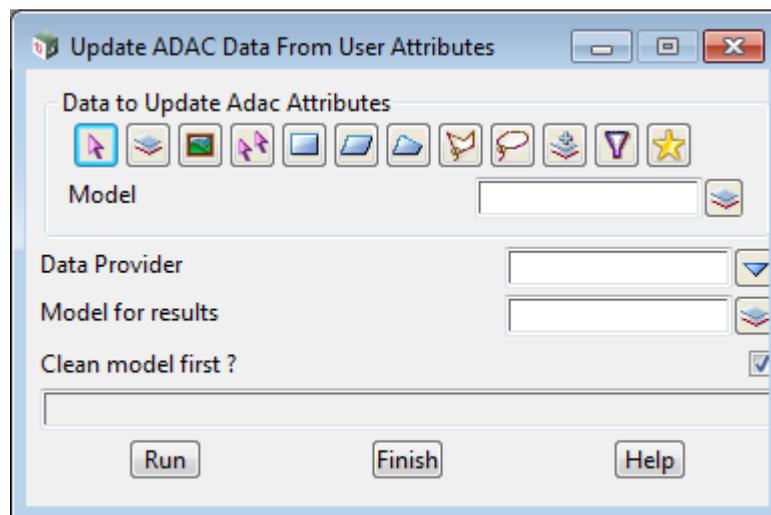
The option is no longer used and had been replaced by a 12d supplied **12duaf file**. See [27.4.8.5 Create/Edit User Attributes to ADAC File](#) and [27.4.8.6 Apply User Attributes to ADAC Elements](#).

There are a number of 12d options, mainly in the ADAC =>User =>Data prep menu, that create User attributes that are to pass data to ADAC Assets. These 12d created user attributes update the ADAC Assets by running this option with the **Data Provider** set to **12d**.

Similarly a user can crate their own user attributes by running their own *12dPLs*, or collecting attributes in the field, or even by hand editing and entering them.

However because these attributes are not know known to *12d Solutions*, the user must set up a **Data Provider** table so the option knows what user attributes to use and what ADAC Assets to update.

Selecting **Set ADAC attributes from user attributes** brings up the **Update ADAC Data from User Attributes** panel.



**Data source type**

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

**Data source**

input

*source of data to process.*

**Data Provider**

choice box

available data providers

*selects the table of User Attributes to use.*

**Model for results**

model box

available models

*model that the processed strings are added to.*

**Clean model first ?**

tick box

*if **ticked**, the model **Model for results** is cleaned before any strings are added to it. If **not ticked**, the model **Model for results** is **NOT** cleaned.*

**Run**

button

*update the ADAC Asset attributes with values from User attributes.*

*If the string has been marked as an ADAC Asset then it is copied and then for the copied string, any relevant ADAC elements updated from the user attributes.*

*Strings that are not marked as ADAC Assets are NOT copied*

*The copied strings are added to the model **Model for results**.*

Continue to [27.4.9.3.6.5 Set ADAC ObjectID from String UID](#) or return to [27.4.9.3.6 12dPL's in Chains](#).

### 27.4.9.3.6.5 Set ADAC ObjectID from String UID

#### Position of option on menu:

File I/O =>ADAC =>User =>Utilities =>12dPLs in chains =>Set ADAC ObjectID from UID

#### From the ADAC XML Schema, the Objectid

*Represents a place for an object identifier, usually generated by the data creation software. Also useful as a placeholder for record ID generated by back-end databases or GIS*

*The Objectid is permitted to be nil because the capturing system may not have the capacity to generate one. If features are exported from a GIS or Asset Management system, however, it is highly recommended to carry them out into this element.*

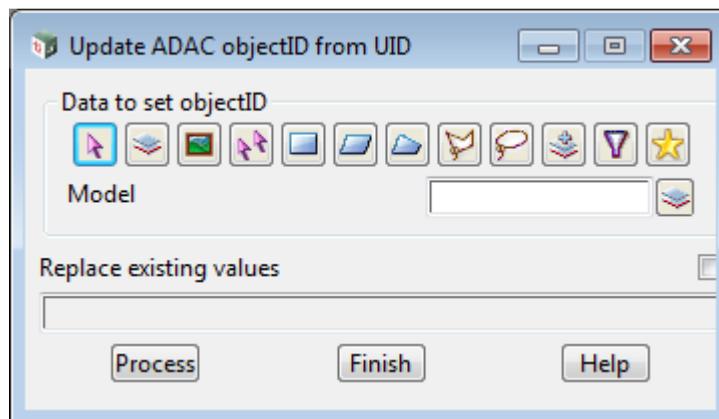
Every string in **12d Model** has a unique identifier called a UID, so **12d Model** creates a unique Objectid by:

- (a) for drainage and sewer strings, creating Objectids for
  - (i) each pit/manhole. The Objectid is made up of the string UID, the pit/manhole vertex index and information about the to help connect the string together.
  - (ii) each pipe. The Objectid is made up of the string UID, the pipe segment index and information about which pits are at the ends of the pipe.

This is only relevant for designers and not surveyors.

- (b) for all other strings, setting the Objectid to the string UID.

Selecting **Set ADAC ObjectID from UID** brings up the **Update ADAC objectID from UID** panel:



#### Data source type

Model

*data selection type - for a full description go to [4.19.3 Data Source](#).*

#### Data source

input

*source of data to process.*

#### Replace existing values

tick box

*if ticked, the model **Model for results** is cleaned before any strings are added to it.  
If not ticked, the model **Model for results** is NOT cleaned.*

#### Process

button

*update the ADAC Asset attributes with values from the 12d string's properties.*

*If the string has been marked as an ADAC Asset then it is copied and then for the copied string, any relevant ADAC elements updated from the properties of the 12d string and the tin.*

*Strings that are not marked as ADAC Assets are NOT copied*

*The copied strings are added to the model **Model for results**.*

In the *ADAC Survey* and *Design* chains, this option is called and arguments are passed from the chain for all the panel fields and so the option runs in the chain without needing to display a panel.

See [27.5.1.2.6 Update ADAC ObjectId - Survey](#) and [27.5.2.2.7 Update ADAC ObjectId - Design](#).

Return to [27.4.9.3.6 12dPL's in Chains](#).

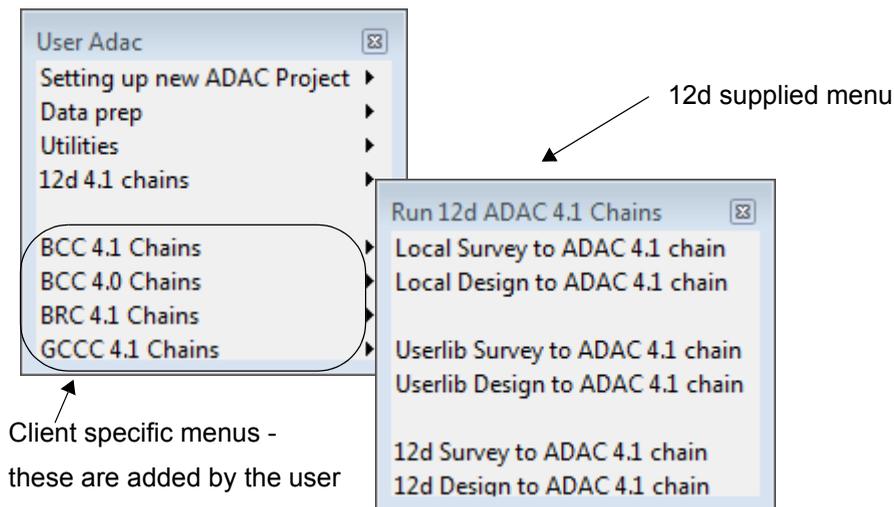
## 27.4.9.4 12d 4.1 Chains

**Position of menu:** File I/O =>ADAC =>User =>12d 4.1 chains

There is a base *ADAC Design* chain and a base *ADAC Survey* chain that does all the work of taking **12d Model** strings and producing ADAC Assets ready to be validated, reported on, or written out to an ADAC XML file.

**12d Model** provides options on the walk right menu **Run 12d ADAC 4.1 Chains** to run the base *ADAC Survey* or *Design* chains for ADAC 4.1 with specially named Map and *12duaf* files in either:

- (a) the working folder for the project (local)
- (b) User\_Lib
- (c) Library.



See

[27.4.9.4.1 Local Survey to ADAC 41 chain](#)

[27.4.9.4.2 Local Design to ADAC 41 chain](#)

[27.4.9.4.3 Userlib Survey to ADAC 41 chain](#)

[27.4.9.4.4 Userlib Design to ADAC 41 chain](#)

[27.4.9.4.5 Lib Survey to ADAC 41 chain](#)

[27.4.9.4.6 Lib Design to ADAC 41 chain](#)

### Note: Client Specific ADAC Design and Survey Menus

The above options are to make it easy to get up and running with ADAC when you have only one or two different Map files.

However, different Clients may do some things differently. For example each company may have its own survey and/or design naming convention or different Authorities may require different information in the ADAC Assets.

For these situations, you can add your own options to **User Adac** menu.

The user options can run either the ADAC Base Survey or Design chains with specific chain pvf files.

For information on setting up your own options on the **User ADAC** menu, see [27.4.9.5 Client Specific ADAC Design & Survey Menus](#).

### 27.4.9.4.1 Local Survey to ADAC 41 chain

**Position on menu:**

File I/O =>ADAC =>User =>12d 4.1 chains =>Local Survey to ADAC 41 chain

The option **Local Survey to ADAC 41 chain** runs *ADAC\_Survey\_Base* chain but with a pvf file that looks in the **working folder for the project** (local) for the files:

***ADAC\_Local\_Map\_Survey\_to\_ADAC\_41.mapfile***

***Local\_41.12duaf*** *this file is optional*

and uses the a **tin** called **ground** to get surface levels.

So to use this menu item, you only need to create the two files and have them in the folder containing the project. This means they are only available for that project.

Continue to [27.4.9.4.2 Local Design to ADAC 41 chain](#) or return to [27.4.9.4 12d 4.1 Chains](#).

### 27.4.9.4.2 Local Design to ADAC 41 chain

**Position on menu**

File I/O =>ADAC =>User =>12d 4.1 chains =>Local Design to ADAC 41 chain

The option **Local Design to ADAC 41 chain** runs *ADAC\_Design\_Base Base* chain but with a pvf file that looks in the **working folder for the project** (local) for the files:

***ADAC\_Local\_Map\_Design\_to\_ADAC\_41.mapfile***

***Local\_41.12duaf***

and uses the a **tin** called **ground** to get surface levels.

So to use this menu item, you only need to create the two files and have them in the folder containing the project. This means they are only available for that project.

Continue to [27.4.9.4.3 Userlib Survey to ADAC 41 chain](#) or return to [27.4.9.4 12d 4.1 Chains](#).

### 27.4.9.4.3 Userlib Survey to ADAC 41 chain

**Position on menu:**

File I/O =>ADAC =>User =>12d 4.1 chains =>Userlib Survey to ADAC 41 chain

The option **Userlib Survey to ADAC 41 chain** runs *ADAC\_Survey\_Base* chain but with a pvf file that looks in the User\_Lib folder for the files:

***ADAC\_Userlib\_Map\_Survey\_to\_ADAC\_41.mapfile***

***Userlib\_41.12duaf*** *this file is optional*

and uses the a **tin** called **ground** to get surface levels.

So to use this menu item, you only need to create the two files and have them in the User\_Lib folder. This means they can be used with any project.

**Note** the word Userlib and not User\_Lib in the name of the Map and 12duaf files.

Continue to [27.4.9.4.4 Userlib Design to ADAC 41 chain](#) or return to [27.4.9.4 12d 4.1 Chains](#).

### 27.4.9.4.4 Userlib Design to ADAC 41 chain

**Position on menu:**

File I/O =>ADAC =>User =>12d 4.1 chains =>Userlib Design to ADAC 41 chain

The option **Userlib Design to ADAC 41 chain** runs *ADAC\_Design\_Base* chain but with a pvf file that looks in the User\_Lib folder for the files:

***ADAC\_Userlib\_Map\_Design\_to\_ADAC\_41.mapfile***

***Userlib\_41.12duaf*** *this file is optional*

and uses the a **tin** called **ground** to get surface levels.

So to use this menu item, you only need to create the two files and have them in the User\_Lib folder. This means they can be used with any project.

**Note** the word Userlib and not User\_Lib in the name of the Map and 12duaf files.

Continue to [27.4.9.4.5 Lib Survey to ADAC 41 chain](#) or return to [27.4.9.4 12d 4.1 Chains](#).

### 27.4.9.4.5 Lib Survey to ADAC 41 chain

**Position on menu:**

File I/O =>ADAC =>User =>12d 4.1 chains =>12d Survey to ADAC 41 chain

The option **12d Survey to ADAC 41 chain** runs *ADAC\_Survey\_Base* chain but with a pvf file that looks in the Library folder that is installed with **12d Model** for the files:

**ADAC\_12d\_Map\_Survey\_to\_ADAC\_41.mapfile**

**12d\_41.12duaf**

*this file is optional*

and uses the a tin called **ground** to get surface levels.

**WARNING:** This option is for demonstration and training purposes only because the files will be overwritten whenever a new version of **12d Model 11** is installed.

**Note** the word 12d and not Library or Lib in the name of the Map and 12duaf files.

Continue to [27.4.9.4.6 Lib Design to ADAC 41 chain](#) or return to [27.4.9.4 12d 4.1 Chains](#).

### 27.4.9.4.6 Lib Design to ADAC 41 chain

**Position on menu:**

File I/O =>ADAC =>User =>12d 4.1 chains =>Lib Design to ADAC 41 chain

The option **12d Design to ADAC 41 chain** runs *ADAC\_Design\_Base* chain but with a pvf file that looks in the Library folder that is installed with **12d Model** for the files:

**ADAC\_12d\_Map\_Design\_to\_ADAC\_41.mapfile**

**12d\_41.12duaf**

*this file is optional*

and uses the a tin called **ground** to get surface levels.

**WARNING:** This option is for demonstration and training purposes only because the files will be overwritten whenever a new version of **12d Model 11** is installed.

**Note** the word 12d and not Library or Lib in the name of the Map and 12duaf files.

Return to [27.4.9.4 12d 4.1 Chains](#).

## 27.4.9.5 Client Specific ADAC Design & Survey Menus

**Position of menu:** File I/O =>ADAC =>User =>Client chains

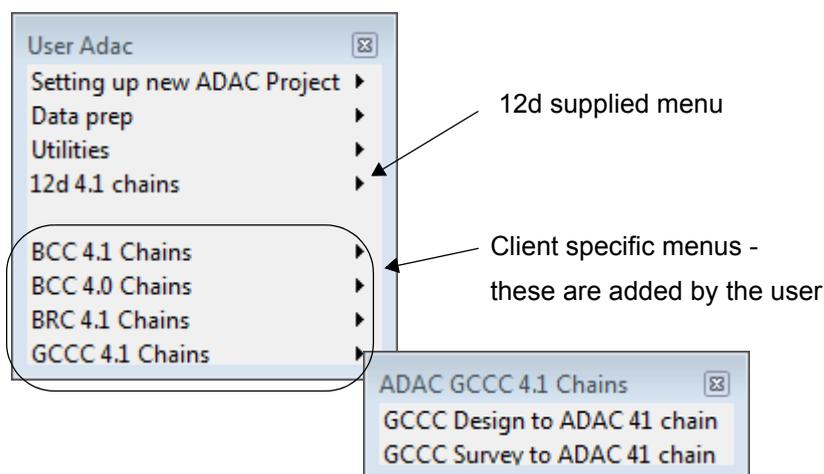
There is a base ADAC Design chain and a base ADAC Survey chain that does all the work of taking **12d Model** strings and producing ADAC Assets ready to be validated, reported on, or written out to an ADAC XML file.

However different Clients may do things differently. For example some Clients may have their own survey or design naming convention and some Clients may need ADAC 4.1 data whilst others want ADAC 4.0.

To allow for such variations, the base ADAC Design chain and Survey chains have parameters passed down to them via a chain pvf file. And you can add your **own** items to the **User Adac** menu that use the base ADAC Survey or Design chains with different chain pvf files.

For these situations, you can add your own options to **User Adac** menu.

A **User ADAC** menu with extra Client specific items would then look like:



How to set up these user defined ADAC menus for your company is described in [27.6.3.2 Setting Up Your User ADAC Menu](#).

### Note: 12d Supplied ADAC 4.1 Design and Survey Menu

If you only have one or two different Map files, **12d Model** supplies options on the walk right menu **Run 12d ADAC 4.1 Chains** of the option **12d 4.1 chains**, to run the base *ADAC Survey* or *Design* chains for ADAC 4.1 with specially named Map and *12duaf* files in either the working folder for the project (local), User\_Lib or Library.

For information on the 12d supplied walk-right menu **Run 12d ADAC 4.1 Chains**, see [27.4.9.4 12d 4.1 Chains](#).

Return to [27.4.9.3 User ADAC Utilities](#).

## 27.5 ADAC Design and Survey Chains

The full explanation of the ADAC Design and ADAC Survey chains is usually only needed for the one or two people in a company who are **setting up** the **ADAC procedures** in the company.

Other users do not need to read it but may find it interesting to read.

In the steps outlined in 12d ADAC Workflow (see [27.3 12d ADAC Workflow](#)) it states that the *ADAC Design and ADAC Survey chains*:

(a) Assign the ADAC Asset Type

Any data going out to ADAC must be one of the ADAC Assets.

The assignment may already have been done by the user with the **Create ADAC Asset** option (see [27.4.2 Create ADAC Asset](#)) but it can also be done automatically using a **12d Map File**. This ADAC Map File must have already been set up for the company.

(b) Sets ADAC attribute values from the 12d String Geometry and User Attributes

Much of the required ADAC data is already contained within the 12d strings or in User Attributes.

For example, if the Drainage or Sewer was designed with **12d Model** then much of the ADAC data can come directly from the drainage and sewer strings. For example, maintenance holes and chambers, depths, and various pipe dimensions are used, required 2D and 3D lengths and areas are calculated.

User attributes picked up in the field by the surveyors, or added in the Data Prep Step, are also loaded into the required ADAC attributes.

(c) Creates the ADAC Geometry

For each ADAC Asset, the ADAC Geometry is automatically generated from the 12d string geometry.

For example, if the ADAC Asset has Polyline Geometry, the (x,y,z) coordinates for each vertex of the string are loaded into the ADAC Geometry.

To go through the workings of the ADAC Design and ADAC Survey chains, see

[27.5.1 Survey to ADAC Chains](#)

[27.5.2 Design to ADAC Chains](#)

## 27.5.1 Survey to ADAC Chains

The **ADAC Survey Chain** takes all the string on the view **Survey to map to adac** and produces ADAC Asset strings in a model **survey adac data** which is added to the view **Survey ready for ADAC**.

So the user simply adds all the strings that are to go into the ADAC XML file onto the view called **Survey to map to adac** and clicks on the appropriate button on a walk right menu on the **User ADAC** menu, to run the chain for a particular customer and ADAC version.

There is actually no **Survey to ADAC Chain** and what the menu option does is run the chain **ADAC\_survey\_base.chain** with a particular chain parameter value file (**pvf** file).

So to fully understand the process, you first need to look at the **pvf** file for the chain **ADAC\_survey\_base.chain**, and then examine in detail the **ADAC\_survey\_base.chain** itself.

See

[27.5.1.1 ADAC Survey Base Chain pvf File](#)

[27.5.1.2 ADAC Survey Base Chain](#)

### 27.5.1.1 ADAC Survey Base Chain pvf File

The **pvf** file for the **ADAC\_survey\_base** chain passes **nine** parameters to the chain - **five** that the user must set and four that are constructed from these five parameters.

1. The text parameter **company** which has as its value an abbreviated customer name. For example, **BCC**.  
This is used to build up the name of the **ADAC 12d Map File** to use.
2. The text parameter **adac\_version\_by\_ten**, which has the value **41** if you are generating ADAC 4.1.0 files, or **40** if you are generating ADAC 4.0 files.  
This is used to build up the name of the **ADAC 12d Map File** to use.
3. The text parameter **user\_attributes\_conversion\_type**
4. The text parameter **survey\_finished\_tin**
5. The text parameter **design\_finished\_tin** is not used in the **ADAC\_survey\_base** chain and can be left blank.

There are two parameters to define which **12d Map Files** to use but they are fully determined once **company** and **adac\_version\_by\_ten** have values.

6. The text parameter **survey\_map\_file** has the value  
\$USER\_LIB\ADAC\_[company]\_Map\_Survey\_to\_ADAC\_[adac\_version\_by\_ten].mapfile
7. The text parameter **design\_map\_file** has the value  
\$USER\_LIB\ADAC\_[company]\_Map\_Design\_to\_ADAC\_[adac\_version\_by\_ten].mapfile  
**design\_map\_file** is not actually used in the **ADAC\_survey\_base** chain.

There are two parameters to define which **12d User Adac to ADAC Elements Files** to use and again they are fully determined once **company** and **adac\_version\_by\_ten** have values.

8. The text parameter **user\_adac\_attribute\_file** has the value  
\$USER\_LIB\[company]\_[adac\_version\_by\_ten].12duaf  
This file is different for each company and must exist (it can contain no information).
9. The text parameter **adac\_12d\_attribute\_file** has the value  
\$USER\_LIB\ADAC\_12d\_[adac\_version\_by\_ten].12duaf  
This file is the same for each company and is for user attributes that **12d Solutions** has created.

The **pvf** file is passed to the **ADAC\_survey\_base.chain** by having the chain run as a menu option on one of the walk right menus on the **User ADAC** menu.

```
Menu "ADAC BCC 4.1 Chains" {
  Button "BCC Design to ADAC 41 chain" {
    Command "chain -pvf $USER/BCC/ADAC_BCC_41.pvf $LIB/ADAC_design_base.chain"
  }
  Button "BCC Survey to ADAC 41 chain" {
    Command "chain -pvf $USER/BCC/ADAC_BCC_41.pvf $LIB/ADAC_survey_base.chain"
  }
}
```

**pvf file to set the parameters for customer BCC and ADAC 4.1**

**base chain**

*ADAC\_Survey\_base.chain* is discussed in detail in [27.5.1.2 ADAC Survey Base Chain](#).

## 27.5.1.2 ADAC Survey Base Chain

The chain *ADAC\_survey\_base.chain* takes all the data on the view **Survey to map to adac** and ends up producing ADAC Assets in a model **survey adac data** which is added to the view **Survey ready for ADAC**.

The screenshot shows the 'Create/Edit Chain' dialog box with the following fields and options:

- Chain file: C:\Program Files (x86)\12d\12dmodel\11.00\library\ADAC\_survey\_base.chain
- Parameter value file: (empty)
- Prompt for parameters: (checked)
- Always record parameters: (checked)
- Run chain in interactive mode: (checked)
- Warning prompt: (checked)

The 'Commands' list is as follows:

- Clean out the models used in processing
- Clean survey mapped data
- Clean survey user attributes data
- Clean survey 12d attributes data
- Clean survey adac data
- Apply Map file to create ADAC Feature attributes
- Run macro ADAC\_separate\_assigned\_and\_unassigned\_adac\_data\_panel
- Map File Apply
- Delete model survey unassigned data
- Fit view survey mapped data
- Update ADAC attributes from non drainage/sewer string properties
- Run ADAC update from 12d properties
- Fit view survey 12d properties data
- Update ADAC attributes from 12d created attributes on strings
- Apply User Attributes to Adac Elements
- Fit view survey 12d attributes data
- Update ADAC attributes from user created attributes on strings
- Apply User Attributes to Adac Elements
- Update ADAC ObjectID from string UID
- Run ObjectID update macro
- Update Geometry section of ADAC from string coordinates
- Adac Synchronise Geometry
- Fit view Survey ready for ADAC

Annotations on the right side of the image point to specific commands:

- Clean out the models that the chain will use (points to 'Clean out the models used in processing')
- see [27.5.1.2.1 Separate the Assigned/Unassigned ADAC Data - Survey](#) (points to 'Run macro ADAC\_separate\_assigned\_and\_unassigned\_adac\_data\_panel')
- see [27.5.1.2.2 Apply the Survey Map File](#) (points to 'Apply Map file to create ADAC Feature attributes')
- see [27.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey](#) (points to 'Update ADAC attributes from non drainage/sewer string properties')
- see [27.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey](#) (points to 'Update ADAC attributes from 12d created attributes on strings')
- see [27.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey](#) (points to 'Update ADAC attributes from user created attributes on strings')
- see [27.5.1.2.6 Update ADAC ObjectID - Survey](#) (points to 'Update ADAC ObjectID from string UID')
- see [27.5.1.2.7 Update ADAC Geometry - Survey](#) (points to 'Update Geometry section of ADAC from string coordinates')

### 27.5.1.2.1 Separate the Assigned/Unassigned ADAC Data - Survey

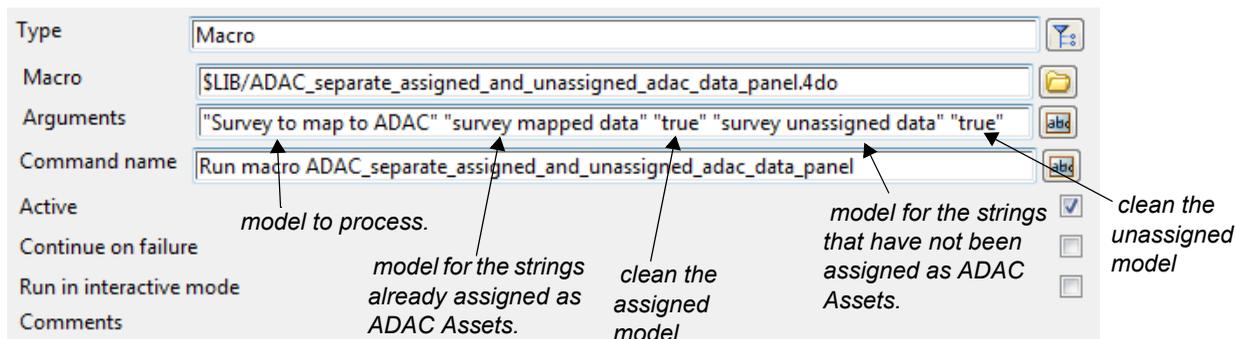
Some strings on the **View** called **Survey to map to ADAC** may have already been assigned (marked) as ADAC Assets and so should not have the ADAC Map file applied to them.

So the data is first processed to separate the assigned and unassigned data into separate models.

This is achieved by running the 12dPL (macro)

```
$LIB/ADAC_separate_assigned_and_unassigned_adac_data_panel.4do
```

which looks at all the data on the **View** called **Survey to map to ADAC** and copies the strings that have been assigned as ADAC Assets to the model **survey mapped data** and all the data that has not yet been assigned to the model **survey unassigned data**.



The model **survey mapped data** is on the view called **survey ready for adac**.

**Note** - this 12d PL program is described in more detail in [27.4.9.3.6.1 Separate Assigned/Unassigned ADAC Data](#)

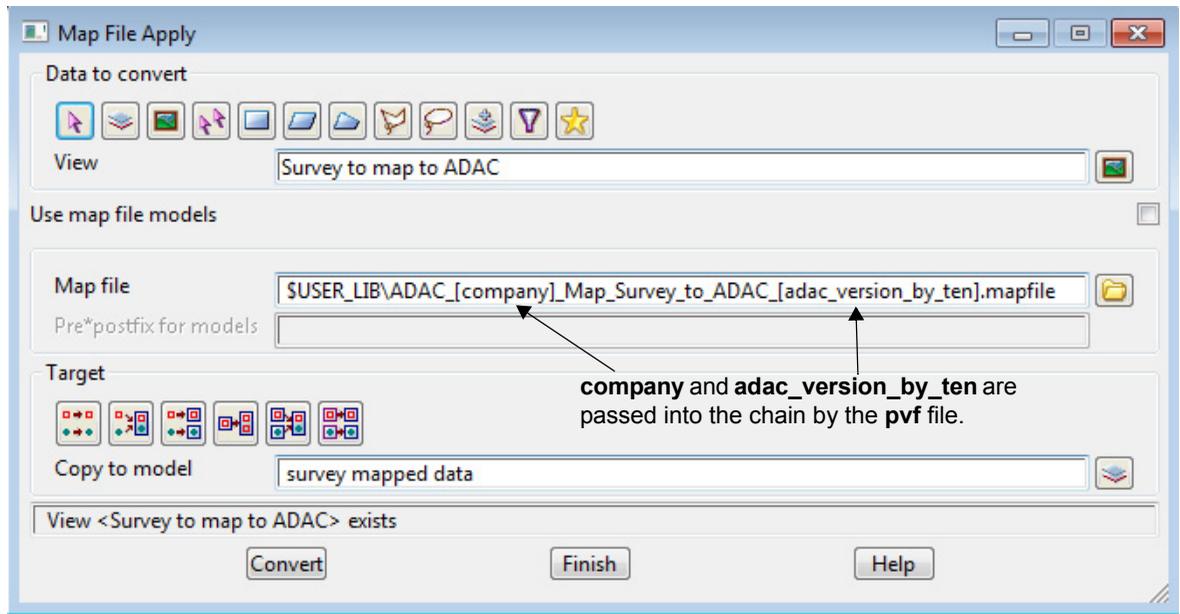
Continue to the next section [27.5.1.2.2 Apply the Survey Map File](#) or return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

### 27.5.1.2.2 Apply the Survey Map File

This section of the chain applies the **12d Map File**

\$USER\_LIB\ADAC\_[company]\_Map\_Survey\_to\_ADAC\_[adac\_version\_by\_ten].mapfile  
to the data in the model **survey unassigned data** and copies all of the string to the model **survey mapped data**.

Most importantly, the **12d Map File** also assigns (marks) some strings as ADAC Assets by giving them ADAC Attributes as defined by the **12d Map File**.



The model **survey mapped data** is on the view called **survey mapped data** and a fit is done on the view.

The model **survey unassigned data** is deleted.

Continue to the next section [27.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey](#) or return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

### 27.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey

Many of the ADAC entities can be calculated directly from the 12d strings and some values may be taken from the tin **survey\_finished\_tin** that is passed into the chain by the **pvf** file.

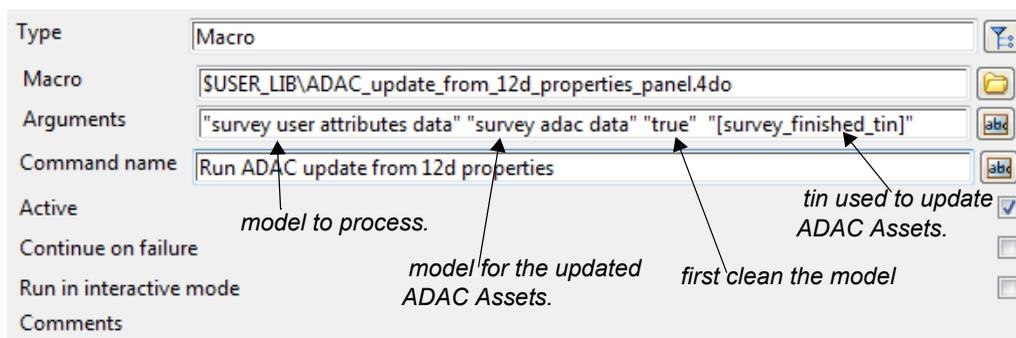
This is achieved by running the 12dPL (macro)

```
$LIB\ADAC_update_from_12d_properties_panel.4do
```

which looks at all the data in the model produced in the previous step and if a string has been marked as an ADAC Asset, then it is copied and any relevant ADAC elements updated from the properties of the 12d string itself and the tin **survey\_finished\_tin**.

The copied string is added to the model **survey adac data**.

Only ADAC Assets are added to the model **survey adac data**.



The model **survey adac data** is on the view called **survey ready for adac**.

**Note** - this 12d PL program is described in more detail in [27.4.9.3.6.2 Set ADAC Attributes from String Properties](#).

Continue to the next section [27.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey](#) or return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

### 27.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey

Some of the values for ADAC elements can be taken from attributes that 12d options have placed on the string. For example, using the ADAC Common Editor panel documented in the section [27.4.9.2.1 Providing Extra Data for ADAC](#).

This is achieved by running the panel

ADAC User Attributes to ADAC Element

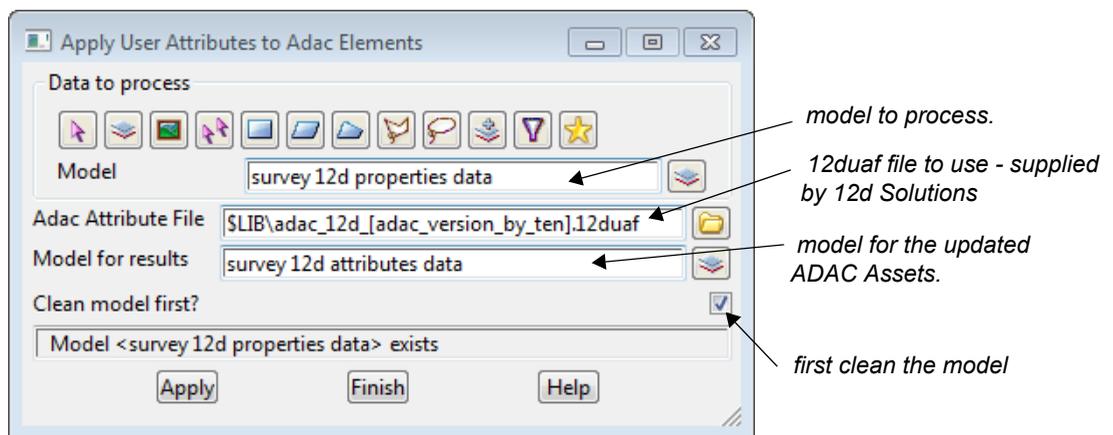
with a *12d Solutions* supplied *12duaf* file for the required version of ADAC.

(this is the option ADAC =>Utilities =>Apply User Attributes to ADAC elements).

The option looks at all the data in the model produced in the previous step and each string is copied and then any relevant ADAC elements updated from User attributes of the string according to the given *12duaf* file.

Note that these User attributes have been created by **12d Model** options that 12d Solutions knows about and hence can provide the *12duaf* file.

The copied strings are added to the model **survey 12d attributes data**.



The model **survey 12d attributes data** is on the view called **survey 12d attributes data** and a fit is done on the view.

**Note** - this option is described in more detail in [27.4.8.6 Apply User Attributes to ADAC Elements](#)

Continue to the next section [27.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey](#) or return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

### 27.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey

Some of the values for ADAC elements can be taken from attributes that users have placed on the string. For example, from attributes picked up by surveyors in the field.

Because these User attributes are defined by the user, *12d Solutions* has no idea of what they are so the user must set up their own *12duaf* file that can be used to update the appropriate ADAC elements from these user created User Attributes.

This user supplied *12duaf* file is placed in the users User\_Lib.

The user supplied *12duaf* file is used by running the panel

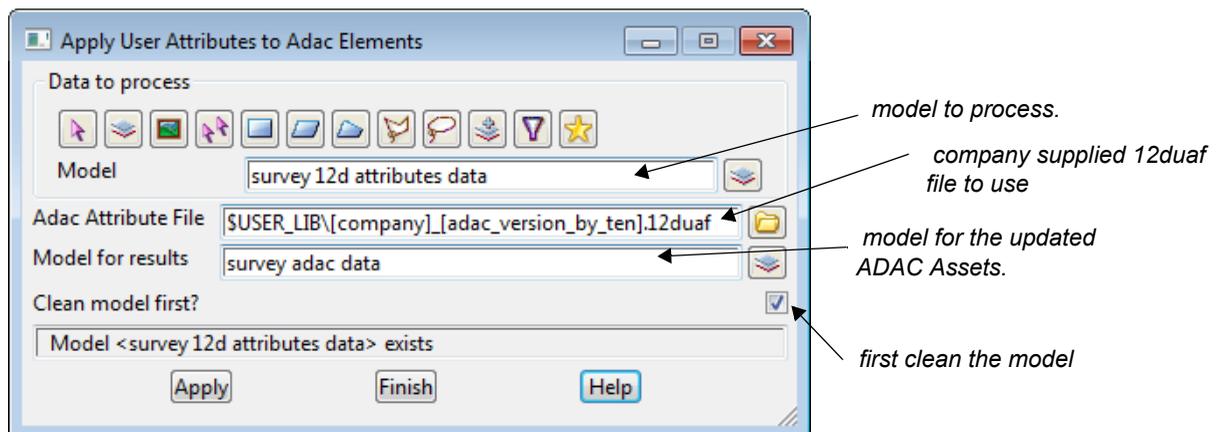
ADAC User Attributes to ADAC Element

with the user supplied *12duaf* file for the required version of ADAC.

(this is the option ADAC =>Utilities =>Apply User Attributes to ADAC elements).

The option looks at all the data in the model produced in the previous step and each string is copied and then any relevant ADAC elements updated from User attributes of the string according to the given *12duaf* file.

The copied strings are added to the model **survey user attributes data**.



The model **survey user attributes data** is on the view called **survey string properties data**.

**Note** - this option is described in more detail in [27.4.8.6 Apply User Attributes to ADAC Elements](#)

Continue to the next section [27.5.1.2.6 Update ADAC ObjectId - Survey](#) or return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

### 27.5.1.2.6 Update ADAC Objectid - Survey

From the *ADAC XML Schema*, the **Objectid**

*Represents a place for an object identifier, usually generated by the data creation software. Also useful as a placeholder for record ID generated by back-end databases or GIS*

*The Objectid is permitted to be nil because the capturing system may not have the capacity to generate one. If features are exported from a GIS or Asset Management system, however, it is highly recommended to carry them out into this element.*

Every string in **12d Model** has a unique identifier called a UID, so **12d Model** creates a unique Objectid by:

- (a) for drainage and sewer strings, creating Objectids for
  - (i) each pit/manhole. The Objectid is made up of the string UID, the pit/manhole vertex index and information about them to help connect the string together.
  - (ii) each pipe. The Objectid is made up of the string UID, the pipe segment index and information about which pits are at the ends of the pipe.

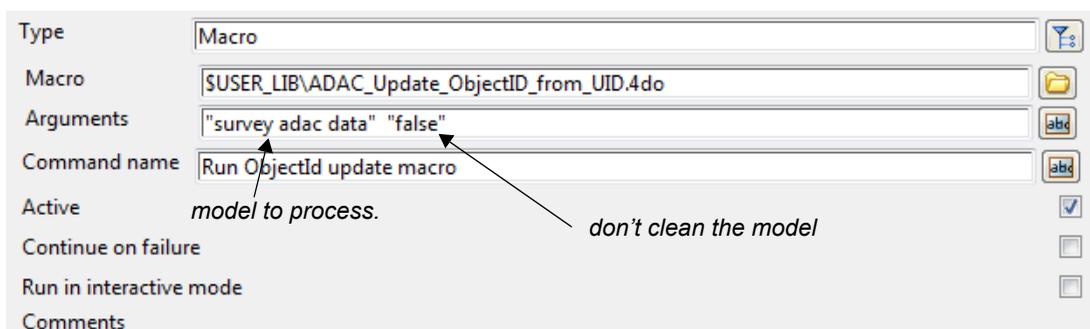
This is only relevant for designers and not surveyors.

- (b) for all other strings, setting the Objectid to the string UID.

The *Objectid* is created by running the 12dPL (macro)

```
$LIB\ADAC_update_Objectid_from_UID.4do
```

The *12dPL* looks at all the strings in the model produced in the previous step, **survey adac data**, and updates the string's own ADAC Objectid attribute. So unlike the previous sections, the string is not copied.



The model **survey adac data** is on the view called **survey ready for adac**.

**Note** - this *12dPL* program is described in more detail in [27.4.9.3.6.5 Set ADAC Objectid from String UID](#).

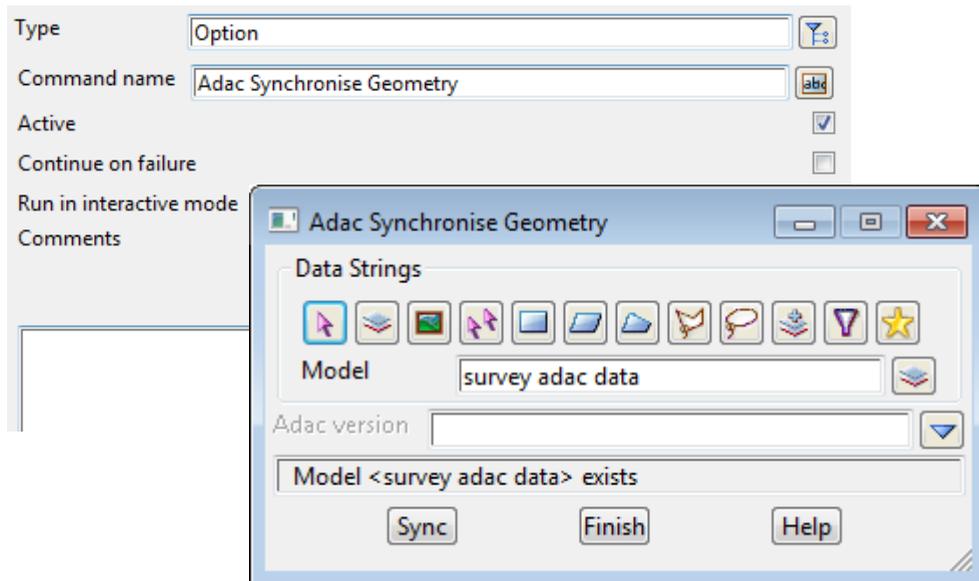
Continue to the next section [27.5.1.2.7 Update ADAC Geometry - Survey](#) or return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

### 27.5.1.2.7 Update ADAC Geometry - Survey

The final step is for every ADAC Asset, updating its ADAC Geometry attributes from the string geometry.

This is done by running in the chain, the option in 12d

**File I/O =>ADAC =>Utilities =>Sync geometry**



The model **survey adac data** is on the view called **survey ready for adac** and a fit is done on the view.

**Note** - this 12d PL program is described in more detail in [27.4.8.4 Sync Geometry](#).

Return to [27.5.1.2 ADAC Survey Base Chain](#) or [27.5.1 Survey to ADAC Chains](#).

## 27.5.2 Design to ADAC Chains

The **ADAC Design Chain** takes all the strings on the view **Design to map to adac** and produces ADAC Asset strings in a model **design adac data** which is added to the view **Design ready for ADAC**.

So the user simply adds all the strings that are to go into the ADAC XML file onto the view called **Design to map to adac** and clicks on the appropriate button on a walk right menu on the **User ADAC** menu, to run the chain for a particular customer and ADAC version.

There is actually no **Design to ADAC Chain** and what the menu option does is run the chain **ADAC\_design\_base.chain** with a particular chain parameter value file (**pvf** file).

So to fully understand the process, you first need to look at the **pvf** file for the chain **ADAC\_design\_base.chain**, and then examine in detail the **ADAC\_design\_base.chain** itself.

See

[27.5.2.1 ADAC Design Base Chain pvf File](#)

[27.5.2.2 ADAC Design Base Chain](#)

### 27.5.2.1 ADAC Design Base Chain pvf File

The **pvf** file for the **ADAC\_design\_base** chain passes **nine** parameters to the chain - **five** that the user must set and four that are constructed from these five parameters.

1. The text parameter **company** which has as its value an abbreviated customer name. For example, **BRC**.

This is used to build up the name of the **ADAC 12d Map File** to use.

2. The text parameter **adac\_version\_by\_ten**, which has the value **41** if you are generating ADAC 4.1.0 files, or **40** if you are generating ADAC 4.0 files.

This is used to build up the name of the **ADAC 12d Map File** to use.

3. The text parameter **user\_attributes\_conversion\_type**
4. The text parameter **survey\_finished\_tin**
5. The text parameter **design\_finished\_tin** is not used in the **ADAC\_survey\_base** chain and can be left blank.

There are two parameters define which **12d Map Files** to use but they are fully determined once **company** and **adac\_version\_by\_ten** have values.

6. The text parameter **survey\_map\_file** has the value  
\$USER\_LIB\ADAC\_[company]\_Map\_Survey\_to\_ADAC\_[adac\_version\_by\_ten].mapfile  
**survey\_map\_file** is not actually used in the **ADAC\_design\_base** chain.
7. The text parameter **design\_map\_file** has the value  
\$USER\_LIB\ADAC\_[company]\_Map\_Design\_to\_ADAC\_[adac\_version\_by\_ten].mapfile

There are two parameters to define which **12d User Adac to ADAC Elements Files** to use and again they are fully determined once **company** and **adac\_version\_by\_ten** have values.

8. The text parameter **user\_adac\_attribute\_file** has the value  
\$USER\_LIB\[company]\_[adac\_version\_by\_ten].12duaf  
This file is different for each company and must exist (it can contain no information).
9. The text parameter **adac\_12d\_attribute\_file** has the value  
\$USER\_LIB\ADAC\_12d\_[adac\_version\_by\_ten].12duaf  
This file is the same for each company and is for user attributes that **12d Solutions** has

created.

The **pvf** file is passed to the **ADAC\_design\_base.chain** by having the chain run as a menu option on one of the walk right menus on the **User ADAC** menu.

```
Menu "ADAC BRC 4.1 Chains" {
  Button "BCC Survey to ADAC 41 chain" {
    Command "chain -pvf $USER/BRC/ADAC_BRC_41.pvf $LIB/ADAC_survey_base.chain"
  }
  Button "BRC Survey to ADAC 41 chain" {
    Command "chain -pvf $USER/BRC/ADAC_BRC_41.pvf $LIB/ADAC_design_base.chain"
  }
}
```

*pvf file to set the parameters for customer BRC and ADAC 4.1*

*base chain*

*ADAC\_Design\_base.chain* is discussed in detail in [27.5.2.2 ADAC Design Base Chain](#)

## 27.5.2.2 ADAC Design Base Chain

The chain *ADAC\_design\_base.chain* takes all the data on the view **design to map to adac** and ends up producing ADAC Assets in a model **design adac data** which is add to the view **Design ready for ADAC**.

**Create/Edit Chain**

Chain file: C:\Program Files (x86)\12d\12dmodel\11.00\library\ADAC\_design\_base.chain

Parameter value file: [Empty]

Prompt for parameters

Always record parameters

Run chain in interactive mode

Warning prompt

**Commands**

- Clean out the models used in processing
- Clean design mapped data
- Clean model design 12d properties data
- Clean design drainage data
- Clean model design 12d attributes data
- Clean design adac data
- Apply Map file to create ADAC Feature attributes
- Run macro ADAC\_separate\_assigned\_and\_unassigned\_adac\_data\_panel
- Run create temporary sewerage attribute macro
- Map File Apply
- Delete model design unassigned data
- Run delete temporary sewerage attribute macro
- Fit design mapped data
- Update ADAC attributes from non drainage/sewer string properties
- Run ADAC update from 12d string properties
- Fit design 12d properties data
- Update ADAC attributes from drainage/sewer string properties
- Run ADAC update drainage data macro
- Fit design drainage data
- Update ADAC attributes from 12d created attributes on strings
- Apply User Attributes to Adac Elements
- Fit view design 12d attributes data
- Update ADAC attributes from user created attributes
- Apply User Attributes to Adac Elements
- Fit Design ready for ADAC
- Update ADAC ObjectID from string UID
- Run objectID update macro
- Update Geometry section of ADAC from string coordinates
- Adac Synchronise Geometry
- Parameters

**Annotations:**

- Clean out the models that the chain will use
- see [27.5.2.2.1 Separate the Assigned/Unassigned ADAC Data - Design](#)
- see [27.5.2.2.2 Applying the Design Map File](#)
- see [27.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design](#)
- see [27.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties](#)
- see [27.5.2.2.5 Update ADAC Elements from 12d Created Attributes on the String - Design](#)
- see [27.5.2.2.6 Update ADAC Elements from User Created Attributes - Design](#)
- see [27.5.2.2.7 Update ADAC Objectid - Design](#)
- see [27.5.2.2.8 Update ADAC Geometry - Design](#)

### 27.5.2.2.1 Separate the Assigned/Unassigned ADAC Data - Design

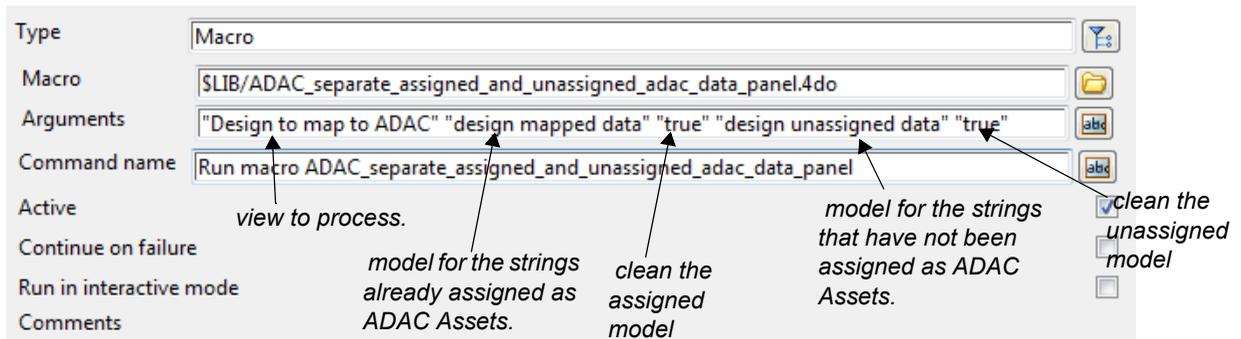
Some strings on the **View** called **Survey to map to ADAC** may have already been assigned (marked) as ADAC Assets and so should not have the ADAC Map file applied to them.

So the data is first processed to separate the assigned and unassigned data into separate models.

This is achieved by running the 12dPL (macro)

```
$LIB/ADAC_separate_assigned_and_unassigned_adac_data_panel.4do
```

which looks at all the data on the **View** called **Design to map to ADAC** and copies the strings that have been assigned as ADAC Assets to the model **design mapped data** and all the data that has not yet been assigned to the model **design unassigned data**.



The model **design mapped data** is on the view called **deign ready for adac**.

**Note** - this 12d PL program is described in more detail in [27.4.9.3.6.1 Separate Assigned/Unassigned ADAC Data](#)

Continue to the next section [27.5.2.2.2 Applying the Design Map File](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.2 Applying the Design Map File

First the 12dPL option *Create\_or\_delete\_temporary\_sewerage\_attribute\_panel.4do* is run on the data in the model called **design unassigned data** and for each

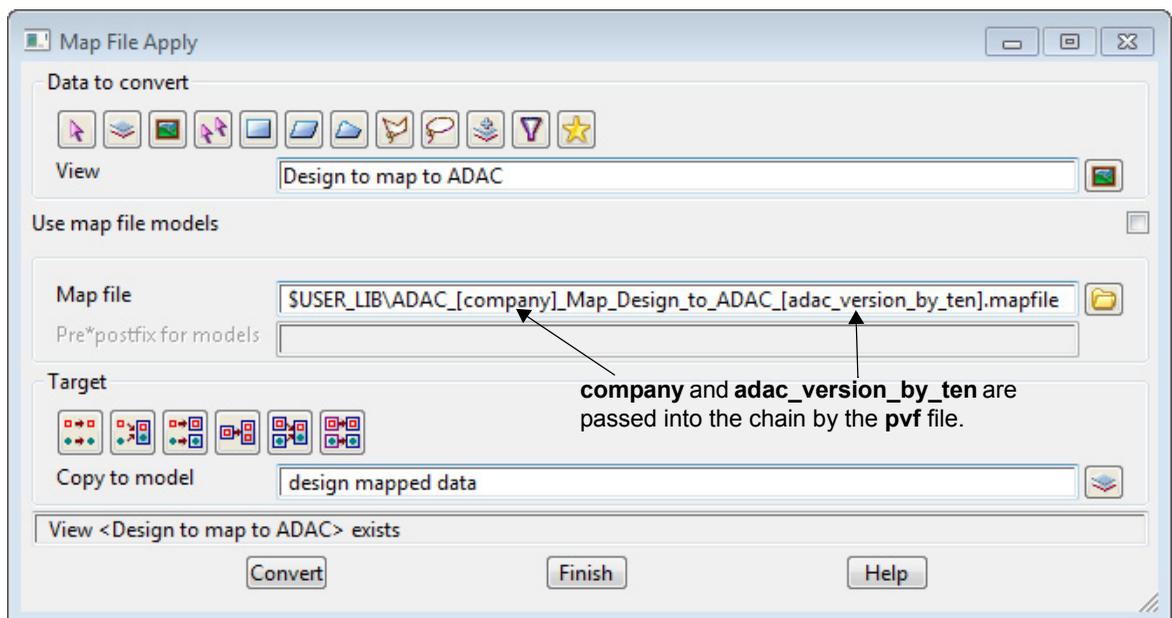
- drainage string**, creates a **string attribute** called **sewertype** of type Integer with 0.
- sewer string**, create a **string attribute** called **sewertype** of type Integer with value 0.

Next the chain applies the **Map File**

\$USER\_LIB\ADAC\_[company]\_Map\_Design\_to\_ADAC\_[adac\_version\_by\_ten].mapfile

to the data on the **View** called **Design to map to ADAC** and copies all of the string and adds to the model **design mapped data**.

Most importantly the **Map File** marks some strings, and the vertices and segments for drainage and sewer strings, as ADAC Assets by giving them ADAC Attributes as defined by the **Map File**.



Next the model **design unassigned data** is deleted.

The 12dPL option *Create\_or\_delete\_temporary\_sewerage\_attribute\_panel.4do* is then run on the data on the view called **Design to map to ADAC** and on the model **design mapped data** to remove the string attribute **sewertype**.

The model **design mapped data** is on the view called **design mapped data** and a fit is done on that view.

Continue to the next section [27.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design

Many of the ADAC entities can be calculated directly from the 12d strings, and some values may be taken from the tin **design\_finished\_tin** that is passed into the chain by the **pvf** file.

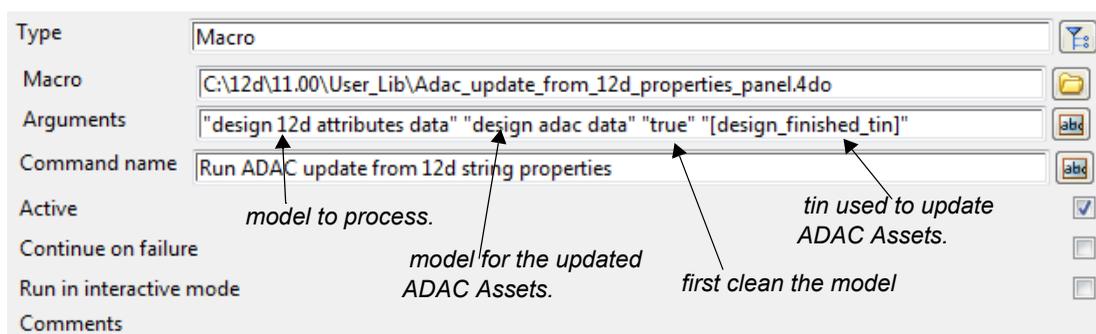
This is achieved by running the 12dPL (macro)

```
$LIB\ADAC_update_from_12d_properties_panel.4do
```

which looks at all the data in the model produced in the previous step and if a string has been marked as an ADAC Asset, then it is copied and any relevant ADAC elements updated from the properties of the 12d string itself and the tin **design\_finished\_tin**.

The copied string is added to the model **design string properties data**.

Only ADAC Assets are added to the model **design string properties data**.



The model **design adac data** is on the view called **design ready for adac**.

**Note** - this 12d PL program is described in more detail in [27.4.9.3.6.2 Set ADAC Attributes from String Properties](#).

Continue to the next section [27.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties

Many of the ADAC entities can be calculated directly from the maintenance holes/pits and pipes from the 12d drainage and sewer strings.

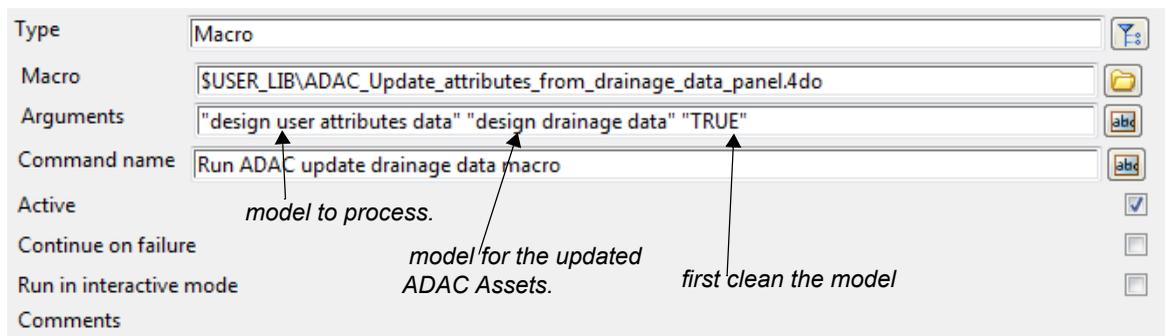
This is achieved by running the 12dPL (macro)

```
$LIB\ADAC_update_from_drainage_data_panel.4do
```

which looks at all the drainage and sewer strings in the model produced in the previous step and if the pits and pipes have been marked as ADAC Assets, then the string is copied and any relevant ADAC pits and pipes in the strings updated from the pits and pipes properties of the 12d string itself.

The copied string is added to the model **design drainage data**.

Only ADAC Assets are added to the model **design drainage data**.



The model **design drainage data** is on the view called **design drainage data** and a fit is done on the view.

**Note** - this 12d PL program is described in more detail in [27.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data](#)

Continue to the next section [27.5.2.2.5 Update ADAC Elements from 12d Created Attributes on the String - Design](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.5 Update ADAC Elements from 12d Created Attributes on the String - Design

Some of the values for ADAC elements can be taken from User attributes that 12d options have placed on the string. For example, using the ADAC Common Editor panel documented in the section [27.4.9.2.1 Providing Extra Data for ADAC](#).

This is achieved by running the panel

ADAC User Attributes to ADAC Element

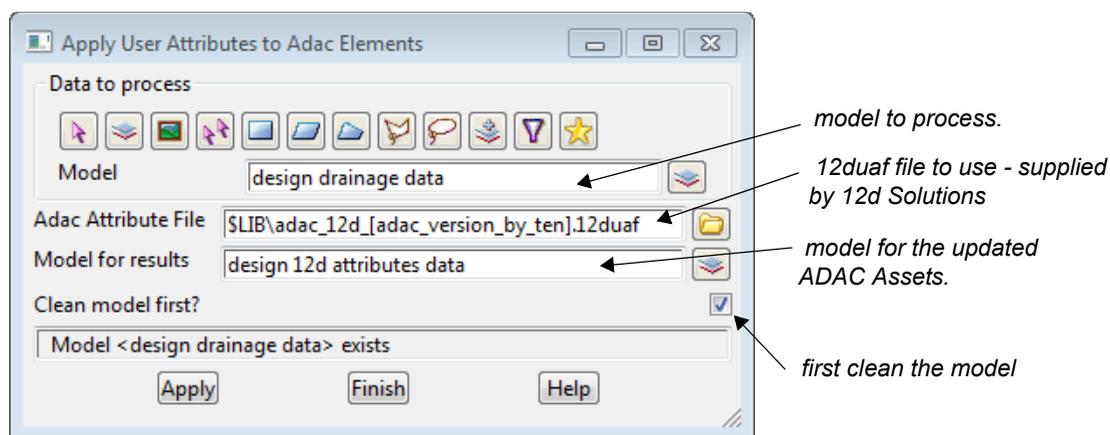
with a 12d Solutions supplied 12duaf file for the required version of ADAC.

(this is the option ADAC =>Utilities =>Apply User Attributes to ADAC elements).

The option looks at all the data in the model produced in the previous step and each string is copied and then any relevant ADAC elements updated from User attributes of the string according to the given 12duaf file.

Note that these User attributes have been created by **12d Model** options that 12d Solutions knows about and hence can provide the 12duaf file.

The copied strings are added to the model **design 12d attributes data**.



The model **design 12d attributes data** is on the view called **design 12d attributes data** and a fit is done on the view.

**Note** - this option is described in more detail in [27.4.8.6 Apply User Attributes to ADAC Elements](#).

Continue to the next section [27.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.6 Update ADAC Elements from User Created Attributes - Design

Some of the values for ADAC elements can be taken from attributes that users have placed on the string.

Because these User attributes are defined by the user, *12d Solutions* has no idea of what they are so the user must set up their own *12duaf* file that can be used to update the appropriate ADAC elements from these user created User Attributes.

This user supplied *12duaf* file is placed in the users User\_Lib.

The user supplied *12duaf* file is used by running the panel

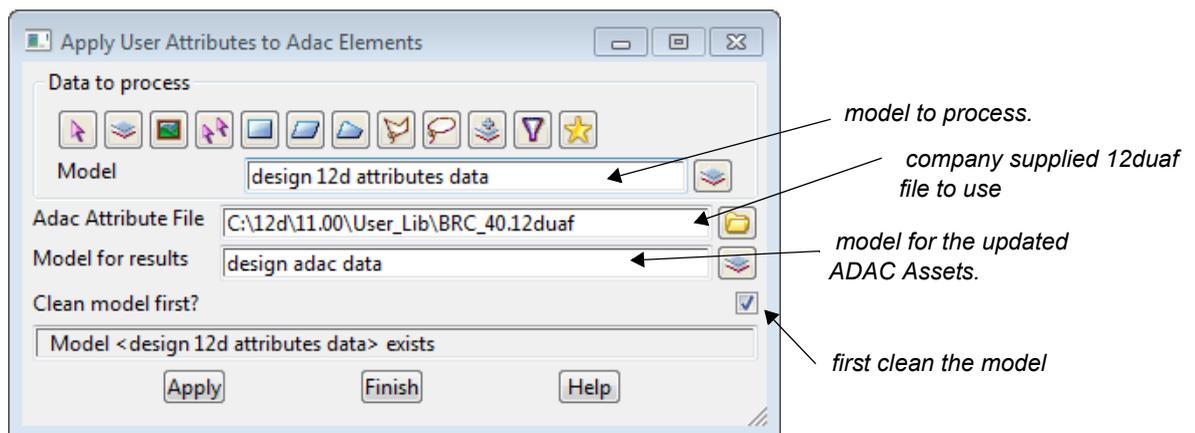
ADAC User Attributes to ADAC Element

with the user supplied *12duaf* file for the required version of ADAC.

(this is the option **ADAC =>Utilities =>Apply User Attributes to ADAC elements**).

The option looks at all the data in the model produced in the previous step and each string is copied and then any relevant ADAC elements updated from User attributes of the string according to the given *12duaf* file.

The copied strings are added to the model **design user attributes data**.



The model **design user attributes data** is on the view called **design string properties data**.

**Note** - this option is described in more detail in [27.4.8.6 Apply User Attributes to ADAC Elements](#)

Continue to the next section [27.5.2.2.7 Update ADAC ObjectId - Design](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.7 Update ADAC ObjectID - Design

From the ADAC XML Schema, the **Objectid**

*Represents a place for an object identifier, usually generated by the data creation software. Also useful as a placeholder for record ID generated by back-end databases or GIS*

*The Objectid is permitted to be nil because the capturing system may not have the capacity to generate one. If features are exported from a GIS or Asset Management system, however, it is highly recommended to carry them out into this element.*

Every string in **12d Model** has a unique identifier called a UID, so **12d Model** creates a unique Objectid by:

- (a) for drainage and sewer strings, creating Objectids for
  - (i) each pit/manhole. The Objectid is made up of the string UID, the pit/manhole vertex index and information about the to help connect the string together.
  - (ii) each pipe. The Objectid is made up of the string UID, the pipe segment index and information about which pits are at the ends of the pipe.

This is only relevant for designers and not surveyors.

- (b) for all other strings, setting the Objectid to the string UID.

The *Objectid* is created by running the 12dPL (macro)

\$LIB\ADAC\_update\_Objectid\_from\_UID\_panel.4do

The 12dPL looks at all the strings in the model produced in the previous step, **design adac data**, and updates the string's own ADAC Objectid attribute. So unlike the previous sections, the string is not copied.

Type	Macro
Macro	\$USER_LIB\ADAC_Update_objectID_from_UID.4do
Arguments	"design adac data" "False"
Command name	Run objectID update macro
Active	<i>model to process.</i>
Continue on failure	<i>don't clean the model</i>
Run in interactive mode	
Comments	

The model **design adac data** is on the view called **design ready for adac**.

**Note** - this 12d PL program is described in more detail in [27.4.9.3.6.5 Set ADAC Objectid from String UID](#).

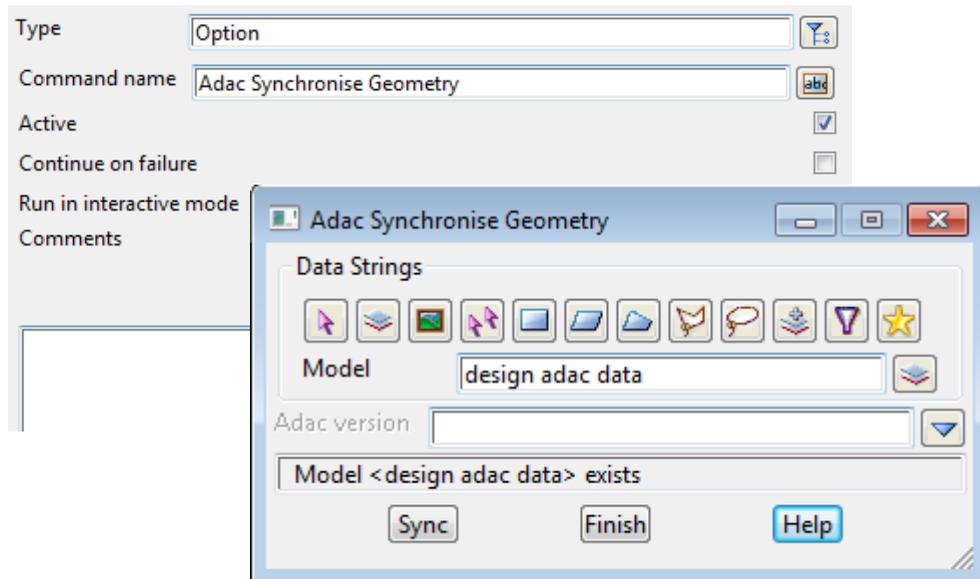
Continue to the next section [27.5.2.2.8 Update ADAC Geometry - Design](#) or return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

### 27.5.2.2.8 Update ADAC Geometry - Design

The final step is for every ADAC Asset, updating its ADAC Geometry attributes from the string geometry.

This is done by running in the chain, the option 12d

**File I/O =>ADAC =>Utilities =>Sync geometry**



The model **design adac data** is on the view called **design ready for adac** and a fit is done on the view.

**Note** - this 12d PL program is described in more detail in [27.4.8.4 Sync Geometry](#)

Return to [27.5.2.2 ADAC Design Base Chain](#) or [27.5.2 Design to ADAC Chains](#).

## 27.6 Setting Up for ADAC

This section is only for the one or two people in a company who are **setting up** the **ADAC procedures** in the company.

Other users should probably avoid reading it.

See

[27.6.1 Setting Up Map Files for ADAC](#)

[27.6.2 What Data Prep is Needed for ADAC](#)

[27.6.3.2 Setting Up Your User ADAC Menu](#)

[27.6.4 Setting Up ADAC Templates](#)

[27.6.5 Setting Up Your User Keys](#)

## 27.6.1 Setting Up Map Files for ADAC

For a **12d Model** string to represent an ADAC Asset, it must have

- (a) string geometry that matches the ADAC Geometry for the ADAC Asset
- (b) as part of its string attributes, the ADAC attribute Group for the particular ADAC Asset.

Picking each string and giving it the correct ADAC attributes for an ADAC Asset can be done using the **ADAC =>Create asset** option and although templates can be used to speed up the process, it is still very time consuming and hence slow.

So in **12d Model**, a **12d Map File** is used to automatically give the appropriate ADAC Asset attributes to strings using the *Attributes* section of the **12d Map File**.

To do this you need to be able to identify each ADAC Asset by a combination of the **string name** and the **string attributes**.

This process is usually much easier for surveyors because when picking up in the field, you could easily set up coding system with a unique string name for each ADAC Asset.

For designers the process can be more difficult and they may need changes to their string naming convention, or some data preparation steps. On the other hand, designers already have most of the information for the ADAC Assets Sewerage>MaintenanceHoles>MaintenanceHole, Sewerage>NonPressurePipes>NonPressurePipe, StormWater>Pits>Pit and StormWater>Pipes>Pipe as part of their drainage and sewer strings and we'll be able to use that without manual intervention.

First we'll look at ADAC in general and then follow on with sections on setting up the ADAC **12d Map File**.

See

[27.6.1.1 Know What ADAC XML Is](#)

[27.6.1.2 Know What Your Client Wants in the ADAC XML File](#)

[27.6.1.3 How the ADAC Map File is Used](#)

[27.6.1.4 Creating the ADAC Attributes for Map Files](#)

[27.6.1.5 Notes On Creating the ADAC Attribute Structure](#)

[27.6.1.6 Setting Up the Map File Selection Criteria](#)

[27.6.1.7 More on Creating the ADAC Map Files](#)

### 27.6.1.1 Know What ADAC XML Is

There is no magic way of being able to set up a **Map File** for ADAC without knowing what the ADAC Assets actually are, what key elements they contain and what Geometry they require.

For example, if you need **WaterSupply>Pipes>Pipe** in the ADAC XML file, then it is critical to know that a **WaterSupply>Pipes>Pipe** can only be a polyline with straight segments. So no arcs.

Or an **OpenSpace>BoatingFacilities>BoatingFacility**, which has **Type** choices of *Jetty*, *Pier*, *Ramp* or *Spillways*, can only be a single **Point**.

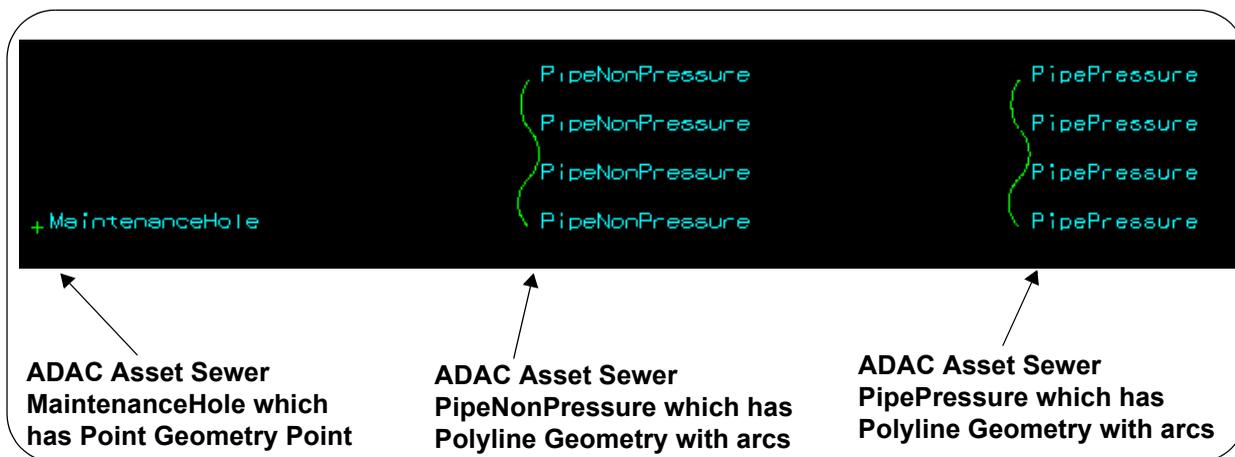
Or that a **Transport>RoadEdges>RoadEdge** must have a **Type** and it can only be one of B1, B2, B3, B4, B5, SM1, SM2, SM3, SM4, SM5, M1, M2, M3, M4, M5, M6, ER1, ER2, ER3, ER4, ER5, INV660, INV90o0, Bitumen and Concrete (as per DMR drawing 1033 - now TMR).

In all cases, any variations from the ADAC Schema will make it an invalid ADAC XML file.

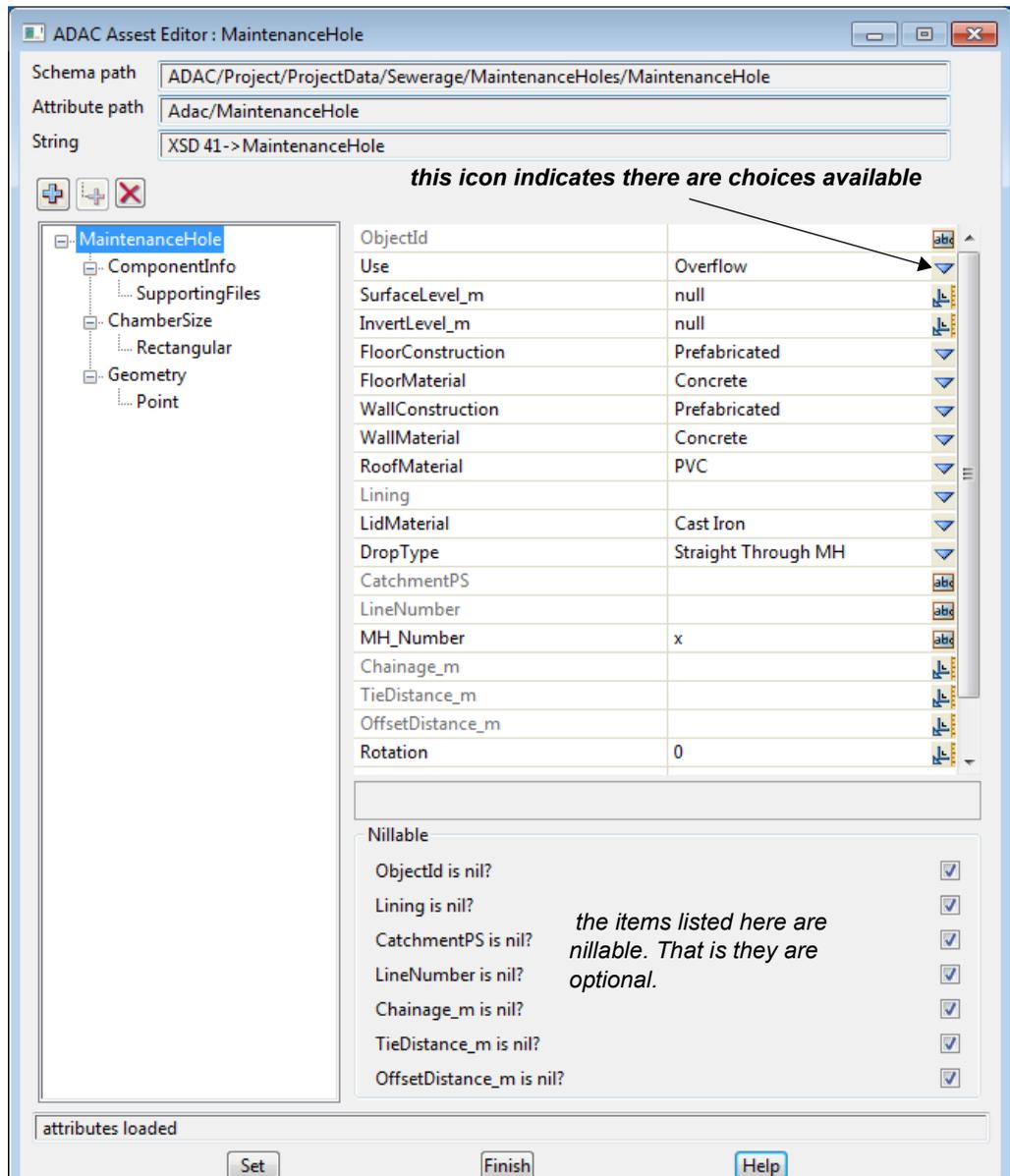
The authorised version of **ADAC Asset Design and As Constructed Schema Help File** (for Version 4.1.0 or Version 4.0.0) is controlled by the **IPWEA** and available on their web site [www.engicom.com.au/products/adac2/](http://www.engicom.com.au/products/adac2/). This is read to learn what the ADAC Assets all are.

To help in the learning process, a very useful **12d** tool is the option **XSD to Model** which takes a given version of the *ADAC XML Schema XSD* and for **each ADAC Asset** in the **XSD**, creates a green super string with the short ADAC Asset name as its string name, for its string geometry an example of the Geometry in the XSD for the ADAC Asset, and in its string attributes, an ADAC group with default values for each of the ADAC element in the ADAC Asset in the XSD. See [27.4.8.3 XSD to Model](#).

For example, the first three strings in that model are:



The 12d string geometry shows graphically what type of Geometry is allowed for that ADAC Asset, and editing any of the strings with the ADAC Edit will display the entire ADAC element structure for the ADAC Asset and see what choices are available, which elements, whether elements are nillable, etc.



Continue to the next section [27.6.1.2 Know What Your Client Wants in the ADAC XML File](#) or return to [27.6.1 Setting Up Map Files for ADAC](#).

## 27.6.1.2 Know What Your Client Wants in the ADAC XML File

The next step is to find out for the Authority you are supplying the ADAC XML file to:

- (a) exactly which ADAC Assets you have to supply in the ADAC XML file
- (b) for each ADAC Asset, what Types/Uses are wanted
- (c) for an ADAC Asset, exactly which elements of the asset they want
- (d) for an ADAC Asset and a compulsory (not nillable) element they do not want, what default value can you put in for it.
- (e) what exactly are the coordinates in the Geometry.

From this you may discover that you only have a small subset of ADAC Assets, and an even smaller subset of the elements of those ADAC Assets, to worry about.

Now that you know exactly what ADAC Assets you need to provide, and what specific elements inside those ADAC Assets, you are ready to set up a 12d Map File to apply to your strings to mark them as particular ADAC Assets.

Continue to the next section [27.6.1.3 How the ADAC Map File is Used](#) or return to [27.6.1 Setting Up Map Files for ADAC](#) .

### 27.6.1.3 How the ADAC Map File is Used

In *12d ADAC*, the **12d Map File** (or just **Map File**) is the major method to create an *ADAC Asset* group of attributes for a string (the process referred to as **marking** the string with an *ADAC Asset*).

The power of this method is that once the **Map File** has been set up to mark all the required strings as *ADAC Assets*, it runs on any number of strings with no manual intervention.

And once set up, the **Map File** is used for every job without modification.

In **12d Model 11**, a new section called **Attributes** was added to the **Map File** and the **12d Map Create/File** panel has a section called **Attributes** which is used to create *String*, *Vertex* and *Segment* attributes for a string.

As with the other sections of the **12d Map Create/File** panel, for each of *Attributes>String*, *Attributes>Vertex* and *Attributes>Segment*, there is a grid.

And for each row in the grid, criteria to check strings being processed against.

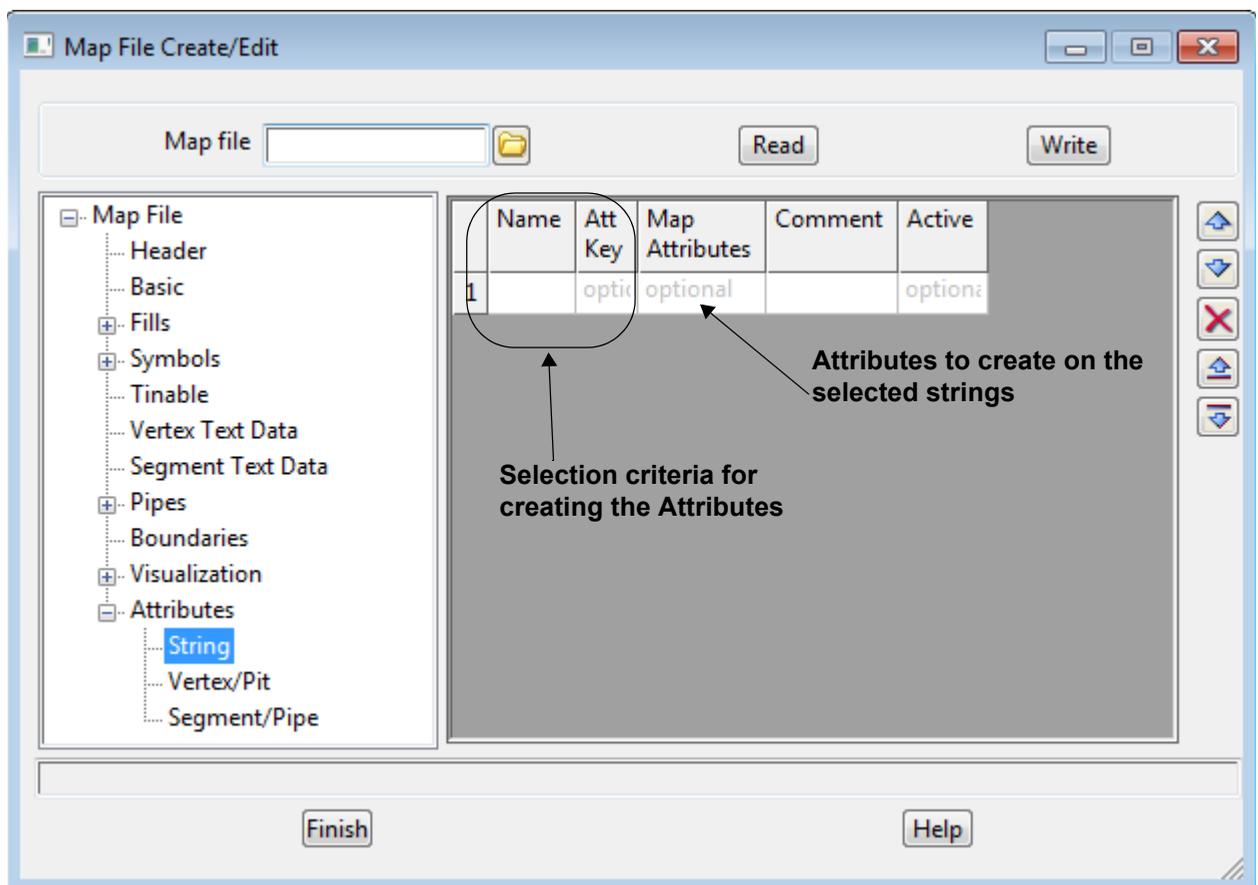
If the criteria are satisfied by a string, then the **Attributes** in the **Map Attributes** column are applied as string/vertex/segment attributes to that string. The next string is then processed.

Once a match to the criteria for a row is made, no more rows of the grid are checked against for that string and the processing starts again for the next string.

The selection criteria for creating *String*, *Vertex* or *Segment* attributes are all different.

#### (a) Creating String attributes

The selection criteria for creating **String** attributes is by **string name** and **string attributes**.



Apart from drainage and sewer strings, this is the only way strings are marked with and *ADAC Asset* by a **Map File**.

The string names are those created during the Design process, or for Surveyors from the field codes when a survey is reduced in **12d Model**.

String attributes can be added by **12d Model** options, 12dPLs (macros), by hand, or picked up in the field by surveyors and brought in the by the survey reduction

(b) Vertex Attributes

The selection criteria for creating **Vertex** attributes is by **string name**, **string attributes** and **vertex attributes**.

In **12d ADAC**, this is only used for processing drainage and sewer strings. Their vertices are often referred to as pits or maintenance holes.

	Name	Att Key	Vertex Att Key	Map Attributes	Comment	Active
1		opti	optional	optional		optiona

**Attributes to create on the selected vertices**

**Selection criteria for creating the Map Attributes**

- string name
- string attribute
- vertex attribute

(c) Segment attributes

The selection criteria for creating **Segment** attributes is by **string name**, **string attributes** and **segment attributes**.

In **12d ADAC**, this is only used for processing drainage and sewer strings. Their segments are often referred to as **pipes**.

	Name	Att Key	Segment Att Key	Map Attributes	Comment	Active
1		opti	optional	optional		optiona

**Attributes to create on the selected segments**

**Selection criteria for creating the Map Attributes**

- string name
- string attribute
- segment attribute

So there are two distinct things that need to be looked at when using a **Map File** to mark strings as ADAC Asset:

- (a) How to set up the ADAC attributes to go in the **Map Attributes** section of the **Map File** that will be applied to a string once it passes the section criteria.

See [27.6.1.4 Creating the ADAC Attributes for Map Files](#)

- (b) How to set up the selection criteria to uniquely determine the ADAC Asset that the string belongs to.

See [27.6.1.6 Setting Up the Map File Selection Criteria](#)

Continue to the next section [27.6.1.4 Creating the ADAC Attributes for Map Files](#) or return to [27.6.1 Setting Up Map Files for ADAC](#).

## 27.6.1.4 Creating the ADAC Attributes for Map Files

The common thing to each of the *Attributes>String*, *Attributes>Vertex* and *Attributes>Segment* grids is that they need sets of ADAC Asset attributes to go in the **Map Attributes** column for each row of the grids.

So we now look at how to best create those ADAC Asset attributes.

You will have noticed when looking at the *ADAC XML Schema Help* File, that its structure is very complex, especially with nullability being used.

And this means that the attribute structure required to replicate the *ADAC Schema* faithfully inside **12d Model** must also be very complex and definitely not the sort of thing you want to do by hand.

So **12d Model** has two powerful tools that simplify the process of creating the ADAC attributes so that the user never has to know the details of how they are stored inside **12d Model**.

### 1. XSD to Map File

The option

**File I/O =>ADAC =>Utilities =>XSD to Map File**

takes the user specified version of the *ADAC XML Schema* XSD and creates a **Map File** with an ADAC group of attributes in the *Attributes>String* section for each ADAC Asset (see [27.4.8.2 XSD to Map File](#) for more details on how the option works).

Although attribute and attribute substructures can be edited, copied, deleted, *etc.* in the *Attributes>String* section of the **Map File**, it is cumbersome when you are dealing with such a complex attribute structure like ADAC.

Also, as a user, you don't want to learn the complex details of how **12d Model** is holding the ADAC data just so that you can make modifications to it.

So although this method is very direct, it is **NOT** recommended.

### 2. Using XSD to Model and ADAC strings to Map File

At first this method may appear more roundabout but it is much more powerful and needs no knowledge of how the ADAC Attributes are stored in **12d Model**.

First the option

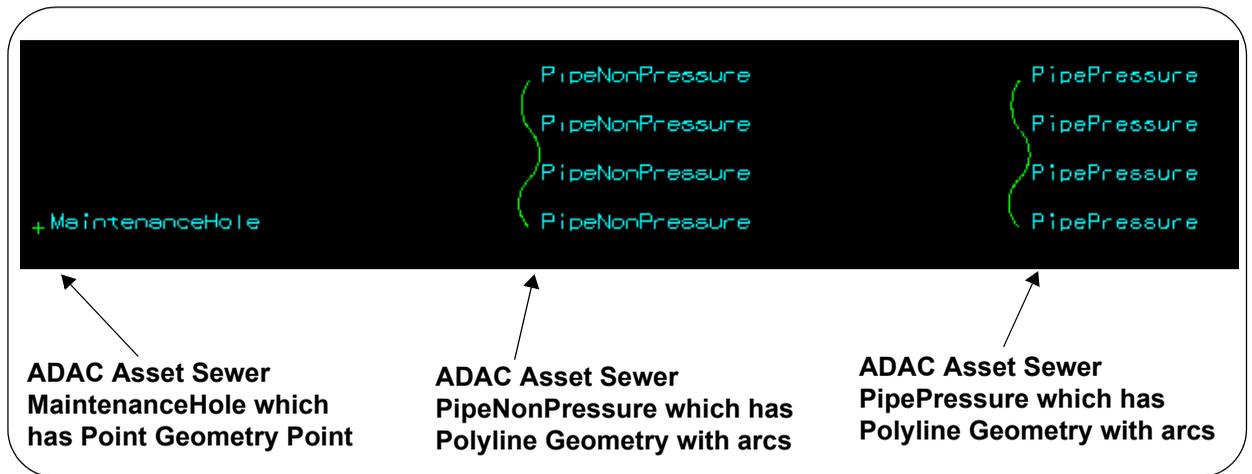
**File I/O =>ADAC =>Utilities =>XSD to model**

is used to take a user specified version of the *ADAC XML Schema* XSD and produces a model with a 12d string for each of the ADAC Assets (so sixty eight strings) and each string has

- (a) a set of ADAC attributes that are typical for that ADAC Asset
- (b) the appropriate string geometry that is allowed for the ADAC Asset Geometry.

See [27.4.8.3 XSD to Model](#) for more information.

For example, running the **XSD to model** option for ADAC XML version 4.1.0 Schema, creates a **model** containing sixty-eight strings and the first three strings in that model are:



You can create an ADAC **12d Map File** from this model by using the option

**File I/O =>ADAC =>Utilities =>ADAC strings to Map File**

and you'll get exactly the same **Map File** that the option **XSD to Map file** produced.

But what we will be doing is creating **new** ADAC Assets in the model by **making a copy of one of the sixty eight**, and then using the **ADAC =>Edit header/asset** option to bring up the **Edit ADAC** panel ([27.4.3 Edit ADAC Header/Asset](#)) and make changes to the copied ADAC Asset.

The benefit of using this editor is that it is totally driven by the **ADAC XML Schema** and so knows all about the ADAC structures and all the ADAC elements.

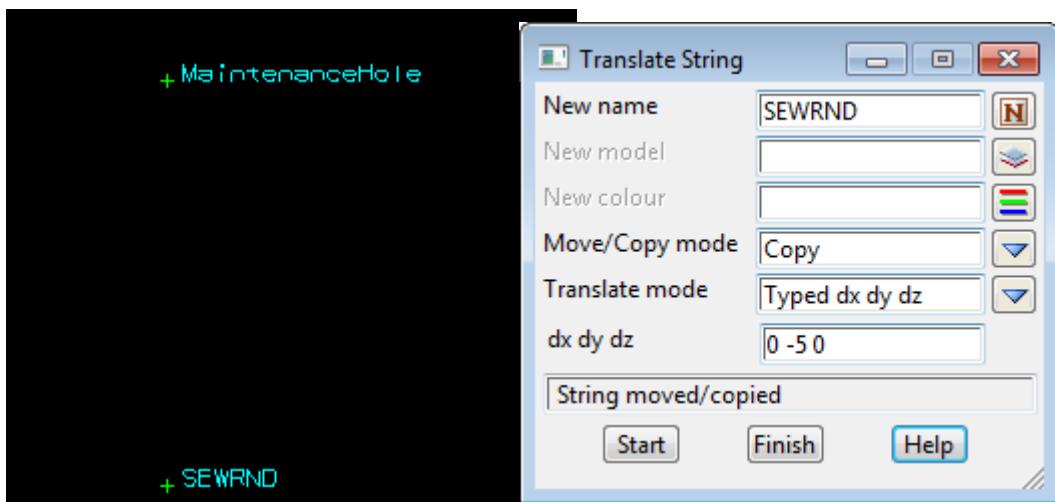
When you run **ADAC Strings to Map File**, each ADAC string creates an entry in either the **Attributes>String**, **Attributes>Vertex/Pit** or **Attributes>Segment/Pipe** section of the **Map File** (see [27.4.8.1 ADAC Strings to Map File](#)).

As an example, the **Sewerage>MaintenanceHoles>MaintenanceHole** created in the model has a rectangular chamber and **Use** is **Overflow**. So we want to create a new MaintenanceHole that has a circular chamber with **Use** as **MaintenanceHole**.

The first step is to copy and translate the existing **Maintenancehole** using

**Strings =>Strings edit =>Translate**

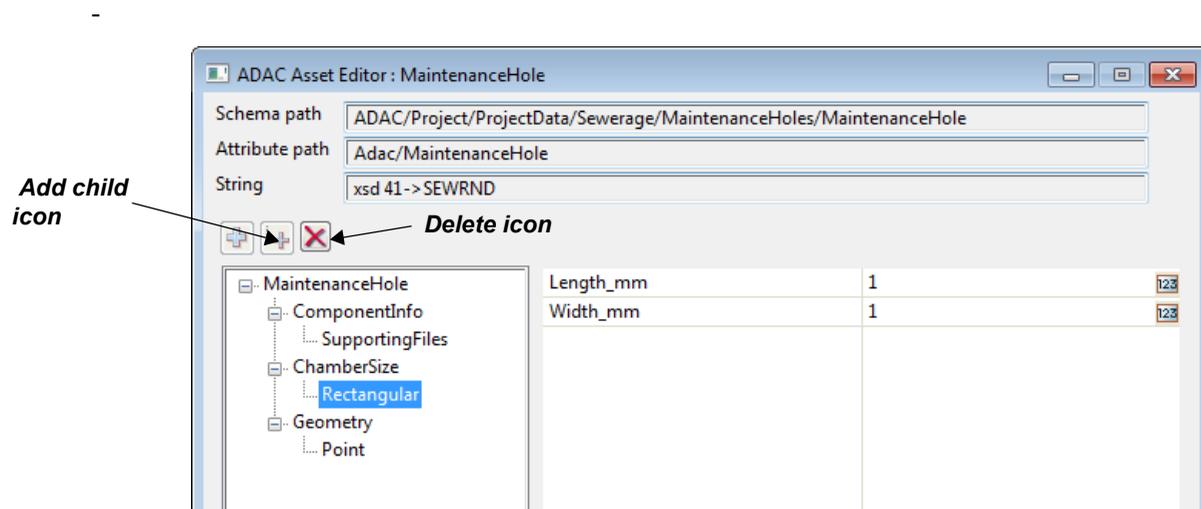
so it isn't on top of the original one, and give the copied string the name **SEWRND**.



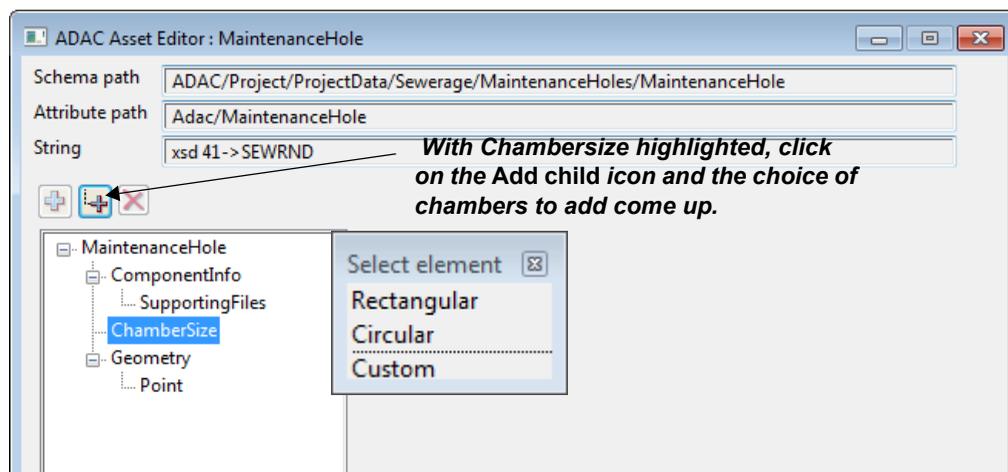
Edit the string SEWRND with the **ADAC =>Edit header/asset** option, and because SEWRND is an ADAC Asset, the **ADAC Asset Editor** for *Sewerage>MaintenanceHoles>MaintenanceHole* is brought up.

Of course because it is a copy, the ADAC attributes are all identical to the string *MaintenanceHole* which had a *Rectangular* chamber.

To change from *Rectangular* to *Circular* chamber, you need to highlight *Rectangular* and then click on the **Delete** icon.



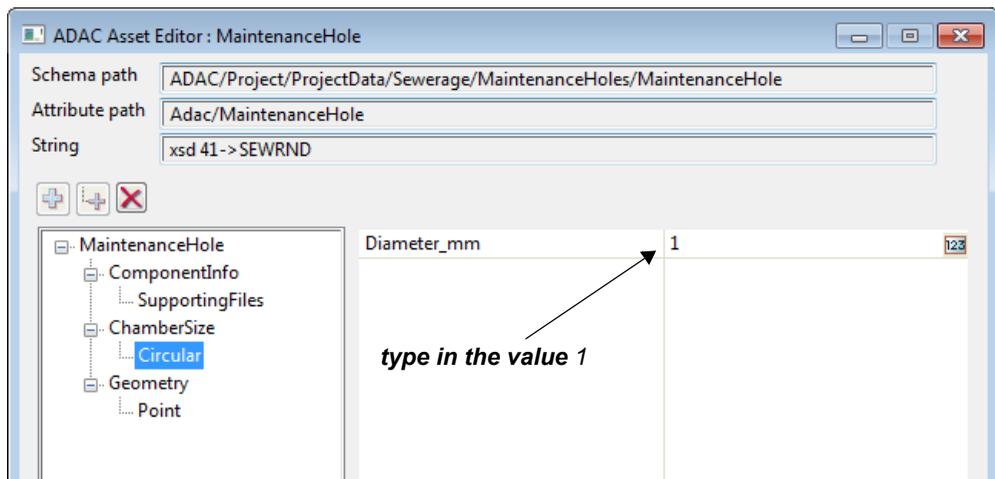
Then click on the **Add Child** icon and select *Circular* from the **Select Element** panel that pops up.



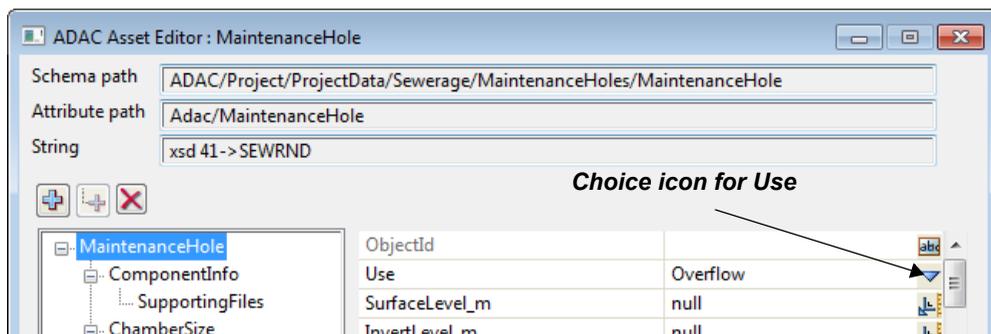
A *Circular* chamber is then added.

A *Circular* chamber has only the one value *Diameter\_mm* and type in a value 1.

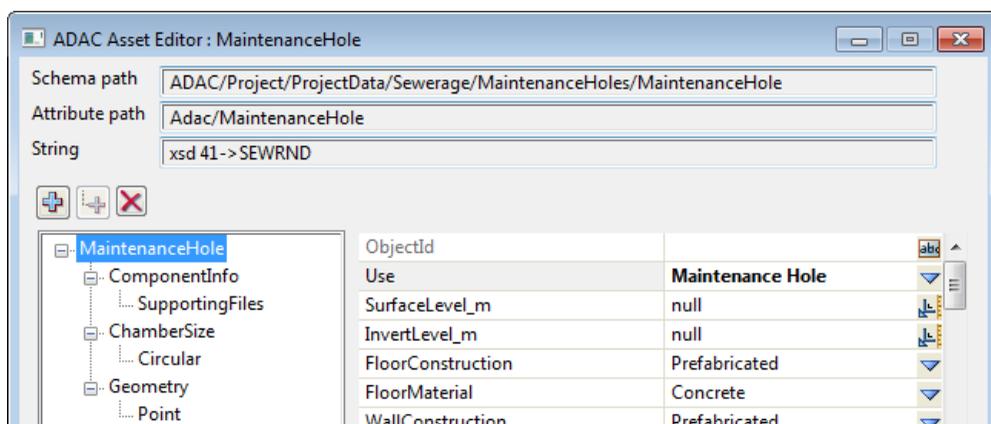
Note that the ADAC Schema says that *Diameter\_mm* is a positive integer, so you must have a valid value in the field or you get an error message when you move to another node.



Similarly, click on the node **MaintenanceHole** and you'll get the first level of ADAC attributes for **MaintenanceHole** displayed on the right hand side, and you'll see that **Use** is one of them with the value *Overflow*.



Click on the **Choice** icon for **Use** and choose *MaintenanceHole* from the list.

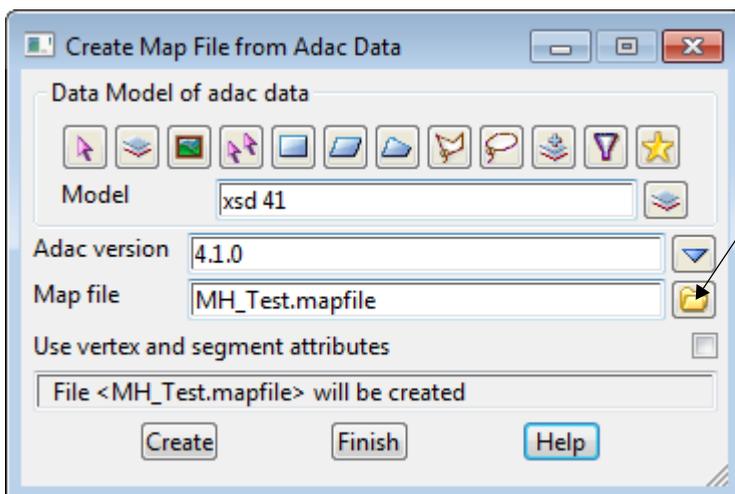


Clicking on **Set** will save the new ADAC attributes to SEWRND.

We will now run the option

**File I/O =>ADAC =>Utilities =>ADAC strings to Map File**

and select the model containing the SEWRND string, the Map file name as **MH\_Test** and the tick box **Use vertex and segment attributes** not ticked:



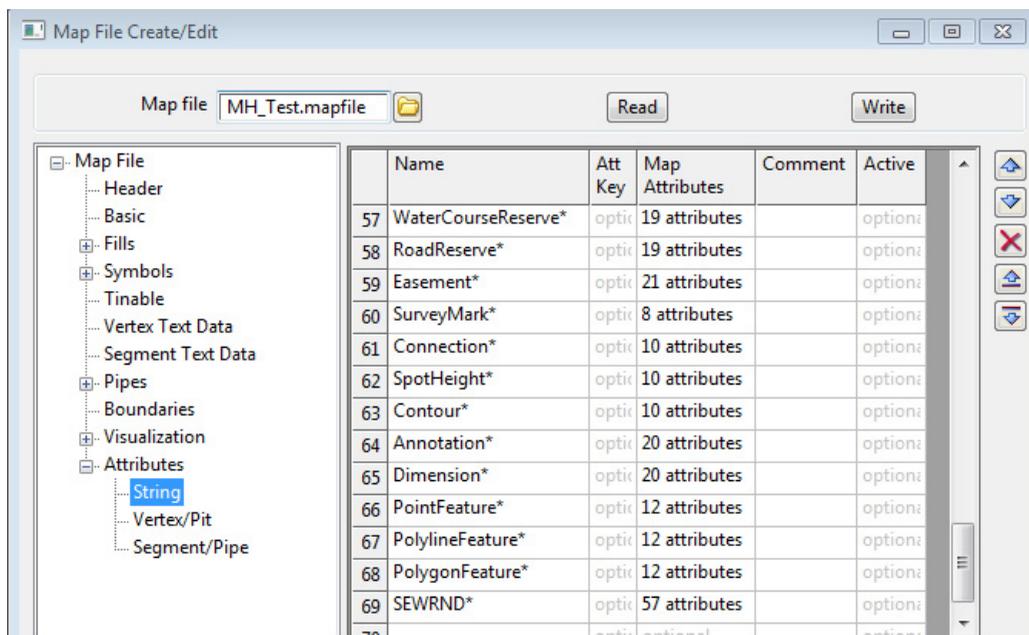
click on the folder icon once the mapfile is created and select Open to edit the Map File

Clicking **Create** produces the **Map File** *MH\_Test.mapfile*.

**Don't** click on **Finish**.

After creating the **Map File**, clicking on the folder icon to the right of the **Map file** field and selecting **Open** will bring up the **Map File** *MH\_Test* in the **Map File Create/Edit** panel.

Click on **Attributes>String** and scroll to the bottom of the grid and you'll see the entry for SEWRND.



**Note:** the grid is filled in the order that the strings are added to the model and SEWRND was the last one added. If needed, the row can be moved up by clicking on the **up** arrow icon on the right hand side.

This **Map file** is now set up with ADAC Asset attributes in the **Map Attributes** column.

Note

If this map file is applied to a string starting with **SEWRND** then the string will be given the ADAC Asset attributes for **Sewerage>MaintenanceHoles>MaintenanceHole** and it has a circular chamber and **Use MaintenanceHole**.

Continue to the next section [27.6.1.5 Notes On Creating the ADAC Attribute Structure](#) or return to [27.6.1 Setting Up Map Files for ADAC](#).

### 27.6.1.5 Notes On Creating the ADAC Attribute Structure

1. Although the process just outlined produced an entry in the **Map File** for SEWRND, the **more important thing** is that it **created** an **ADAC Attributes structure** in the *Map Attributes* column.

And we had total control over what was in the ADAC attribute structure simply by using the **ADAC Editor** to set the attributes up exactly as we wanted them without ever needing to know how the attribute structure is stored in **12d Model**.

So this is the process we suggest for creating all the ADAC Attribute structures, even if they are never used with the string name you used to create them.

Permanently keep the model (or models) of the strings with their Attribute Structures so they can be used to generate new **Map Files** at any time, and as examples to copy when you are creating new Attribute Structures in the future.

2. In the example we only created a **Map File** with an *Attributes>String* section, but the panel **Create Map File from ADAC Data** can also produce entries in the *Attributes>Vertex/Pit* and *Attributes>Segment/Pipe* section for special ADAC strings (see [27.4.8.1 ADAC Strings to Map File](#)).

However, the important thing is that the **ADAC Attribute structure** has been created for a *MaintenanceHole*. It can then be copied and pasted to other **Map Attributes** columns as required.

Continue to the next section [27.6.1.6 Setting Up the Map File Selection Criteria](#) or return to [27.6.1 Setting Up Map Files for ADAC](#).

### 27.6.1.6 Setting Up the Map File Selection Criteria

There are three ways we select strings for ADAC in the Attributes section of the **Map File**.

- (a) Use the **string name** only

This is for not for drainage and sewer strings.

See [27.6.1.6.1 Using String Names to Select Strings](#)

- (b) Using the **string name** and/or a **string attribute**

This is for not for drainage and sewer strings.

See [27.6.1.6.2 Using String Name and/or String Attributes to Select Strings](#)

- (c) For drainage and sewer strings only

Using vertices and segments of the drainage and sewer strings and **vertex names** and **segment names**.

See [27.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings](#)

### 27.6.1.6.1 Using String Names to Select Strings

This is only for strings that are not Drainage or Sewer strings

See [27.6.1.6.1.1 Designers' Approach](#)

See [27.6.1.6.1.2 Surveyor Approach - Using Survey Codes in the Field](#)

#### 27.6.1.6.1.1 Designers' Approach

The first question to ask, for a string that is not a drainage or sewer string, is:

for a string in the design, is the **string name** enough to uniquely identify it as a particular ADAC Asset, or even better, identify it as a particular ADAC Asset and one of its choices for **Type** (or **Use**)?

For example, the string name **Kerb** is enough to identify it as being a **Transport>RoadEdges>RoadEdge** but it would not be enough to identify it as a **RoadEdge** of **Type B1** as opposed to being **Type B2** or **Type Bitumen**.

The easiest way of overcoming that problem is to have a **different string name** for each **RoadEdge** and **Type** that is required.

For example, **EB** for RoadEdge of type **Bitumen** (a common one), **EB1** for RoadEdge of Type **B1** and so on.

So where ever possible, try and adopt a string naming convention that has a unique name for each of the ADAC Assets of a particular **Type** (or **Use**) that are required by the Authority.

The **Map File** already created works for this situation - just have a model with a string of each of the names and the attributes that go with it and the **ADAC Strings to Map File** option will create the **Map File**.

#### What if that is not possible and the string naming convention cannot be changed?

For example, the one string name **Kerb** is used for **all** the **RoadEdge** of **Type B1, B2, B3, SM1, etc.**

In that case, it would be necessary to use the **string name** and a **User string attribute**, or **just a User string attribute**. See [27.6.1.6.2 Using String Name and/or String Attributes to Select Strings](#)

#### 27.6.1.6.1.2 Surveyor Approach - Using Survey Codes in the Field

Once you know exactly what ADAC Assets, and what parts of those ADAC Assets, you need information on, can you code the information in the field so that once it is read into **12d Model**, a **Map File** using the string name and if that is not enough, string attributes, to determine which ADAC Asset the string represents?

This assignment is often much easier for surveyors because you may be able to easily add extra field codes to distinguish the different ADAC Assets. And you will know what geometry is required for the ADAC Asset, even though in other circumstances you would have picked it up another way.

For example, for **OpenSpace>BoatingFacilities>BoatingFacility**, it only can have a single **Point**. So if all you are giving is ADAC XML, you only have to pick up **one** point.

Also think about the code being able to determine what the ADAC Asset is, but it may provide more information about the ADAC Asset.

In the **OpenSpace>BoatingFacilities>BoatingFacility** example, a compulsory element is **Type** with the choices of *Jetty, Pier, Ramp* or *Spillways*.

You could set up your codes so not did you know it was a BoatingFacility but also what **Type**.

For example, the codes could be

**BFJ** is an **OpenSpace>BoatingFacilities>BoatingFacility** of Type *Jetty*.

**BFP** is an **OpenSpace>BoatingFacilities>BoatingFacility** of Type *Pier*.

**BFR** is an **OpenSpace>BoatingFacilities>BoatingFacility** of Type *Ramp*.

**BFS** is an **OpenSpace>BoatingFacilities>BoatingFacility** of Type *Spillway*.

The **Map File** already created works for this situation - just have a model with a string of each of the names and the attributes that go with it and the **ADAC Strings to Map File** option will create the **Map File**.

#### **What if that is not possible and the string naming convention can not be changed?**

For example, the one string name **Kerb** is used for **all** the **RoadEdge** of Type B1, B2, B3, SM1, etc.

In that case, it would be necessary to use the **string name and a User string attribute**, or **just a User string attribute**. See [27.6.1.6.2 Using String Name and/or String Attributes to Select Strings](#)

### 27.6.1.6.2 Using String Name and/or String Attributes to Select Strings

A string has three types of *User* defined attributes

- (a) string attributes - there is one set of these for the whole sting
- (b) vertex attributes - there is a set for each vertex
- (c) segment attributes - there is a set for each segment.

However **only string attributes** are used by the ADAC **Map File**.

*User string attributes* can be created by **12d Model** options, 12PLs (macros), manually, or if you are a surveyor, by picking them up in the field.

See

[27.6.1.6.2.1 Entering User Attributes in the Field](#)

[27.6.1.6.2.2 Running a 12d Option or 12dPL that Produce User Attributes](#)

[27.6.1.6.2.3 Manually Entering User Attributes](#)

[27.6.1.6.2.4 Using the Name and Att Key in the ADAC Map File](#)

### 27.6.1.6.2.1 Entering User Attributes in the Field

As well as using field codes, a surveyor may also have the capability of picking up attributes in the field.

An example of when attributes are useful is when picking up a **WaterSupply>Pipes>Pipe** - its diameter may be entered as an attribute.

Or when picking up a **Sewerage>MaintenanceHoles>MaintenanceHole**, which can only have a **Point** for its **Geometry**, you could enter the width and length as attributes.

Note that some other survey software can only produce vertex attributes. So if any of those attributes are to be used with the *Map File*, a method must be found to turn them into string attributes.

If you can do field pick ups with attributes, even if the attributes are not used with the **ADAC Map File**, they may be very useful in the step *Update ADAC Elements from User Attributes* that is further along in the ADAC chain. See [27.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey](#).

Continue to the next section [27.6.1.6.2.2 Running a 12d Option or 12dPL that Produce User Attributes](#) or return to [27.6.1.6.2 Using String Name and/or String Attributes to Select Strings](#).

### 27.6.1.6.2.2 Running a 12d Option or 12dPL that Produce User Attributes

Any **12d Model** options that create *User Attributes* will be document with what those *User attributes* are.

For example, [27.4.9.2.3 Generate ADAC Road Edge Types](#) splits a string into user defined chainage ranges and adds a User string attribute named **RoadEdge\_Type** for each chainage range. That is, the 12dPL spits the Kerb into distinct parts, each with the same name Kerb but each part having a *User string attribute* called *RoadEdge\_Type* which has only one of the **RoadEdge Types** as its value.

So this is a method for transcending the problem of having only the single string name of **Kerb**.

Hopefully any 12dPLs (macros) that you use are also documented, and in that it describes how the 12dPL works and what User string attributes it sets.

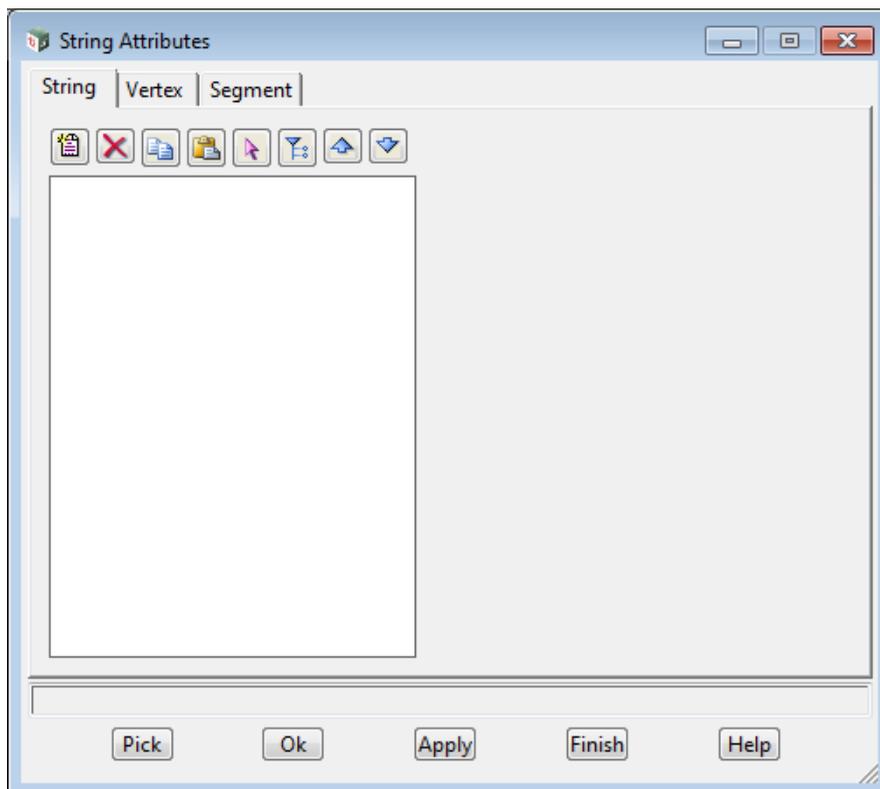
Continue to the next section [27.6.1.6.2.3 Manually Entering User Attributes](#) or return to [27.6.1.6.2 Using String Name and/or String Attributes to Select Strings](#).

### 27.6.1.6.2.3 Manually Entering User Attributes

If you are adding a *User string attribute* manually, (or wanting to display existing attributes for the string), pick the option

**Strings =>Properties =>Attributes**

This brings up the **Strings Attributes** panel



Click on **Pick** and then select the string to add attributes to. Any existing string, vertex or segment attributes are displayed by clicking on the **String**, **Vertex** or **Segment** tabs.

With **String Attributes**, User string attributes of virtually any name and value can be created to distinguish between Types of an ADAC Asset.

But this is a very manual process.

If you need to do it regularly, then it is best to write a small 12d PL (macro) that lets you pick a string, asks for the value for the attribute and then updates the string attribute. This will save lots of time and errors.

*User string attributes* are used to uniquely identify the type of ADAC Asset that a string represents, but they are also very useful for supply values for elements inside an ADAC Asset as well. See [27.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey](#) and [27.5.2.2.6 Update ADAC Elements from User Created Attributes - Design](#).

Continue to the next section [27.6.1.6.2.4 Using the Name and Att Key in the ADAC Map File](#) or return to [27.6.1.6.2 Using String Name and/or String Attributes to Select Strings](#).

#### 27.6.1.6.2.4 Using the Name and Att Key in the ADAC Map File

The **Map File** that is created with the option **ADAC Strings to Map File** automatically creates a **Map File** for matching the string name against the **Name** in the **Map File**.

But when using a **string name** AND a **string attribute**, or **just a string attribute**, then you need to use both the **Name** column and the **Att Key** (Attribute key) column in the grid for the **Attributes>String** of the **Map File**.

Luckily, there is a way of setting up an ADAC string so that the **ADAC string to Map File** option creates both the **Name** column and the **Att Key** column entries - you simply need to add the required **Att Key** attribute with its type and value as the **string attribute** of the **ADAC string**.

So creating the row of the **Map File** when the criteria involves a **string name** and a **string attribute** is a two step process.

##### Step 1. Create the required ADAC attribute structure

This has already been described in the section [27.6.1.4 Creating the ADAC Attributes for Map Files](#).

##### Step 2. Edit the ADAC string and add the Att Key criteria

Here we take the **ADAC string** produced in *Step 1* and manually edit it with the **Strings Attributes** panel and add the required string attribute that will be copied to the **Att Key** column, and hence used as part of the criteria for selecting a string.

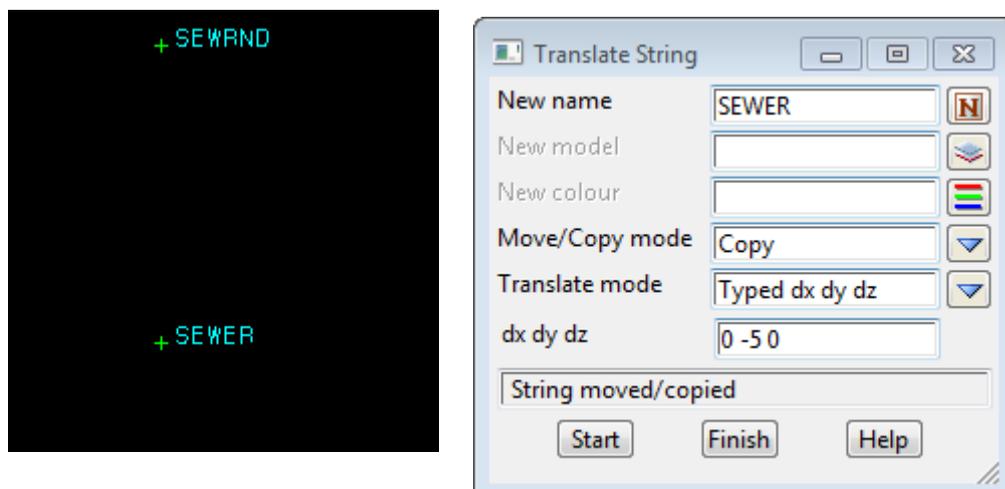
To explain the process of adding a string attribute that becomes an **Att Key**, we will build on the previous example and

Instead of having a **string name** SEWRND to indicate the *Sewerage>MaintenanceHoles>MaintenanceHole* has a circular chamber, we only have the **nondescript name** SEWER and a **string attribute** called **Chamber** which has the text value of **Circular** or **Rectangular**.

To do the **Circular** case, copy and translate the existing ADAC *Sewerage>MaintenanceHoles>MaintenanceHole* called **SEWRND** using the option

**Strings =>Strings edit =>Translate**

and name the copied string **SEWER**.

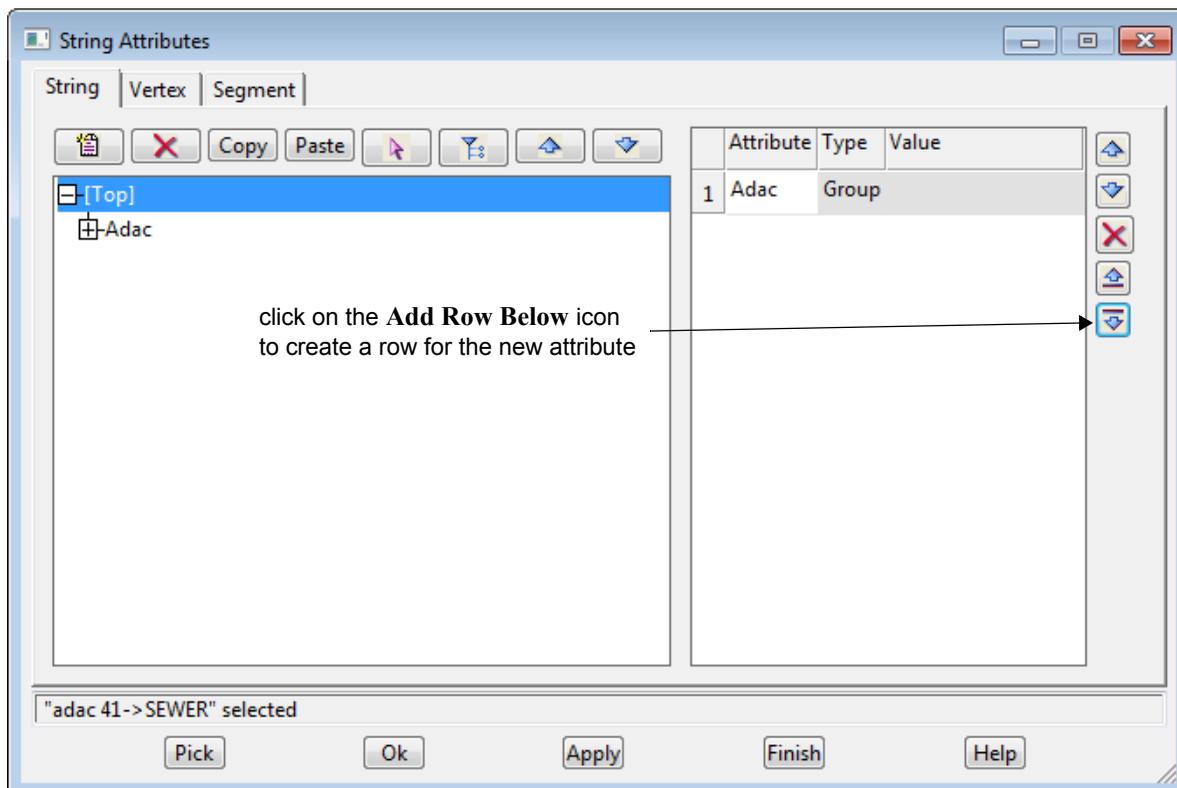


The ADAC string SEWER already has a **Circular** chamber so we now only have to add the text string attribute **Chamber** with the value **Circular**.

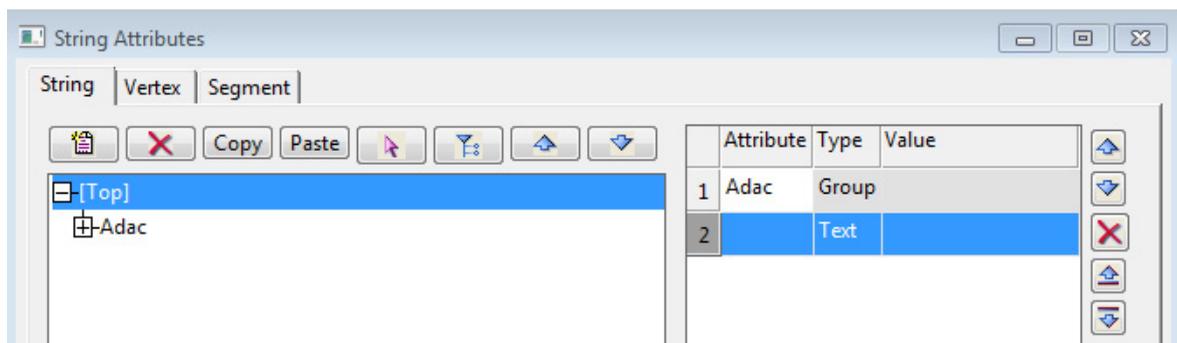
So the add attributes to the SEWER string, bring up **String Attributes** panel selecting the option

**Strings =>Properties =>Attributes**

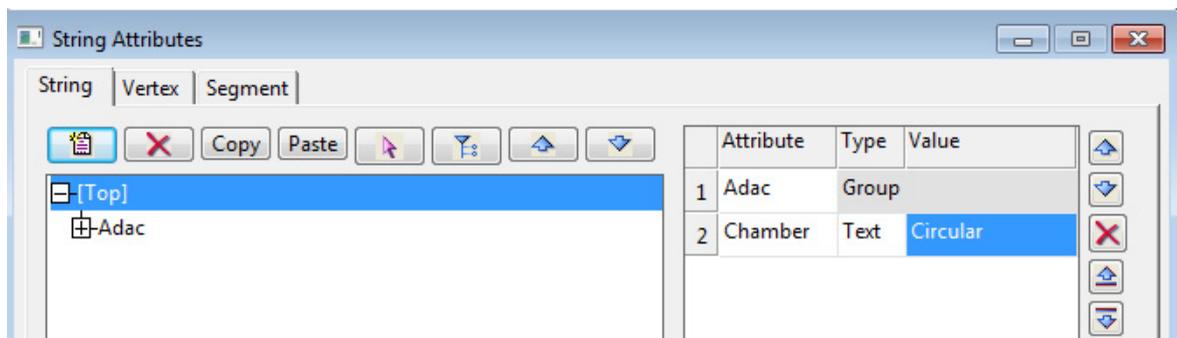
Click on **Pick** and select the string SEWER.



Highlight the node **[Top]** and then click on the **Add Row Below** icon to create a new row to use for the attribute **Circular**.



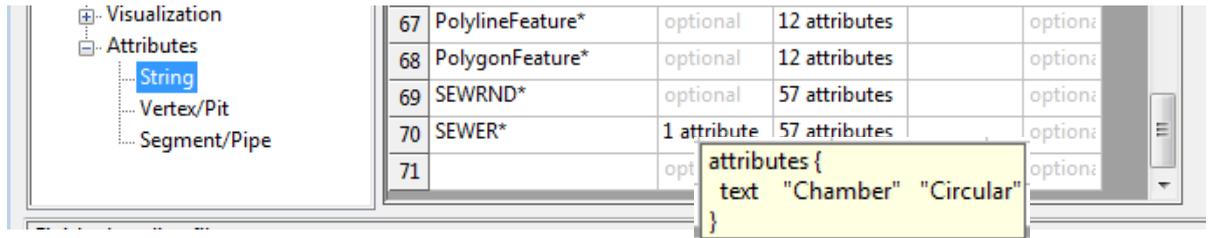
Type in the attribute name **Chamber** and the value **Circular** and then click on the **Apply** button **twice** to add the attribute to the string.



To create the **Map File**, we repeat the process used in [27.6.1.4 Creating the ADAC Attributes for Map Files](#) which is to run the option

**File I/O =>ADAC =>Utilities =>ADAC strings to Map File**

to create the **Map File** and then open the **Map File** and go to the bottom of the *Attributes>String* grid and hover over the **Att Key** column to see that the attribute Chamber is defined.



So this **Map File** will select any string with the string name starting with SEWER and with a text attribute called **Chamber** with the value **Circular**, and give it the ADAC attributes for a *Sewerage>MaintenanceHoles>MaintenanceHole* with a Circular chamber.

As an alternative to having the attribute Chamber as part of the ADAC string SEWER, it is possible to add the Att Key values into the **Map File** by hand. So **Step 2** would be replaced by:

#### **Step 2'.Edit the Map File and add the Att Key criteria**

Here we take the **Map File** produced in *Step 1* and manually edit it with the **Map File Create/Edit** panel and add the required **Att Key** in for the string attribute that is to be used as part of the criteria for selecting a string. This will now be described.

The problem with this approach is this that if the Map File is generated again from the ADAC strings, any manual edits such as in Step 2' will be lost and have to be repeated, or some other more complicated strategy employed such a keeping ADAC strings that require an **Att Key** for identification in a **separate** model.

So it is recommended NOT to use **Step 2'** but to use **Step 2** where a string attribute is added to the to the ADAC string.

Continue to the next section [27.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings](#) or return to [27.6.1.6 Setting Up the Map File Selection Criteria](#) .

### 27.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings

The Designers drainage and sewer strings are unique in that they hold data for four different ADAC Assets and the four ADAC Assets are known straight away.

1. Each **pit** of a Drainage string is a *StormWater>Pits>Pit*.
2. Each **pipe** of a Drainage string is a *StormWater>Pipes>Pipe*
3. Each **pit (maintenance hole)** of a Sewer string is a *Sewerage>MaintenanceHoles>MaintenanceHole*.
4. Each **pipe** of a Sewer string is a *Sewerage>PipesNonPressure>PipeNonPresssure*.

So for drainage or sewer string, each **vertex** and each **segment** of the string needs to be marked as a ADAC Asset. Consequently, unlike all the other strings for ADAC, the **Attributes>Vertex** and **Attributes>Segment** sections of the **Map File** are used.

Unfortunately a **Map File** doesn't know the difference between a drainage and a sewer string as it is an internal property of the string so just before the **Map File** is applied in the *Design Chain*, the 12dPL *Create\_or\_delete\_temporary\_sewerage\_attribute\_panel* is run and for each drainage and sewer string, it creates a **string attribute** called **sewerstype** of type Integer with the value **0** if it is a **drainage string**, or **1** if it is a **sewer string**.

Being a **string attribute**, the attribute **sewerstype** CAN be used in a **Map File**.

So when going through drainage and sewer strings, if it is a

- (i) vertex with an integer string attribute **sewerstype** of value **1** then it is a *Sewerage>MaintenanceHoles>MaintenanceHole*
- (ii) vertex with an integer string attribute **sewerstype** of value **0** then it is a *StormWater>Pits>Pit*
- (iii) segment with an integer string attribute **sewerstype** of value **1** then it is a *Sewerage>PipesNonPressure>PipeNonPresssure*.
- (iv) segment with an integer string attribute **sewerstype** of value **0** then it is a *StormWater>Pipes>Pipe*.

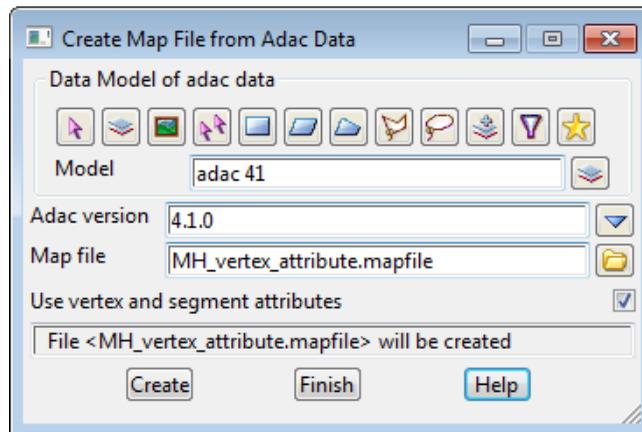
**Important Note** - Actually a **Map File** can't tell the difference between ANY string types, so the presence of a **string attribute** called **sewerstype** with value **0** or **1** can be used in a **Map File** to select only a sewer string or a drainage string.

Knowing this we can now set up the *Attributes>Vertex* and *Attributes>Segment* sections of the **Map File** to mark the vertices and segments of drainage and sewer strings as ADAC Assets

So far we have only used the option

**File I/O =>ADAC =>Utilities =>ADAC strings to Map File**

to create rows in the *Attributes>String* section of the **Map File** but if we click on the tick box **Use vertex and segment attributes** in the **Create Map File from ADAC Data** panel,



then

the ADAC attributes for ADAC Sewerage>MaintenanceHoles>MaintenanceHole's and ADAC StormWater>Pits>Pit's will be placed in the *Attributes>Vertex/Pit* section of the **Map File** with the **Att Key** set with the appropriate **sewerType** attribute value

and

the ADAC attributes for ADAC Sewerage>PipesNonPressure>PipeNonPressure's and ADAC StormWater>Pipes>Pipe's will be placed in the *Attributes>Segment/Pipe* section of the **Map File** with the **Att Key** set with the appropriate **sewerType** attribute value

(For more information on Use vertex and segment attributes in the **Create Map File from ADAC Data** panel see [27.6.1.4 Creating the ADAC Attributes for Map Files](#) )

A string attribute on the ADAC Asset will also go through as a Vertex Att Key or Segment Att Key, and the string name is written as a comment.

For examples of the vertex and the segment cases, see

[27.6.1.6.3.1 Setting Up the Attributes>Vertex Section of the Map File](#)

[27.6.1.6.3.2 Setting Up the Attributes>Segment Section of the Map File](#)

[27.6.1.6.3.3 Bonuses for Drainage and Sewer Strings](#)

### 27.6.1.6.3.1 Setting Up the Attributes>Vertex Section of the Map File

The strings for ADAC Assets that create an entry in the *Attributes>String* section of the *Map File* use the string **name** and **Att Key** as a method of deciding which strings have the *Map Attributes* applied to them.

But the ADAC strings for *Sewerage>MaintenanceHoles>MaintenanceHole* or *StormWater>Pits>Pit* which create an entry in the *Attributes>Vertex/Pit* section of the *Map File*, already have the **Name** column set to \* and the **Att Key** is used with **sewertype** to select only drainage or sewer strings.

So the only part of the *Attributes>Vertex/Pit* row left help to differentiate between different types of maintenance holes/pits, is the **Vertex Att Key**.

Luckily the vertices of **12d Model** drainage and sewer string usually have a text **Vertex attribute** called **pit\_type**, or some other vertex attributes set during creation, and these can be used to select **Map Attributes** that are more appropriate to that vertex.

With that in mind, the **Create Map File from ADAC Data** panel with **Use vertex and segment attributes** ticked **on**, copies the string attribute of the ADAC Asset *Sewerage>MaintenanceHoles>MaintenanceHole* or *StormWater>Pits>Pit*, to the **Vertex Att Key**.

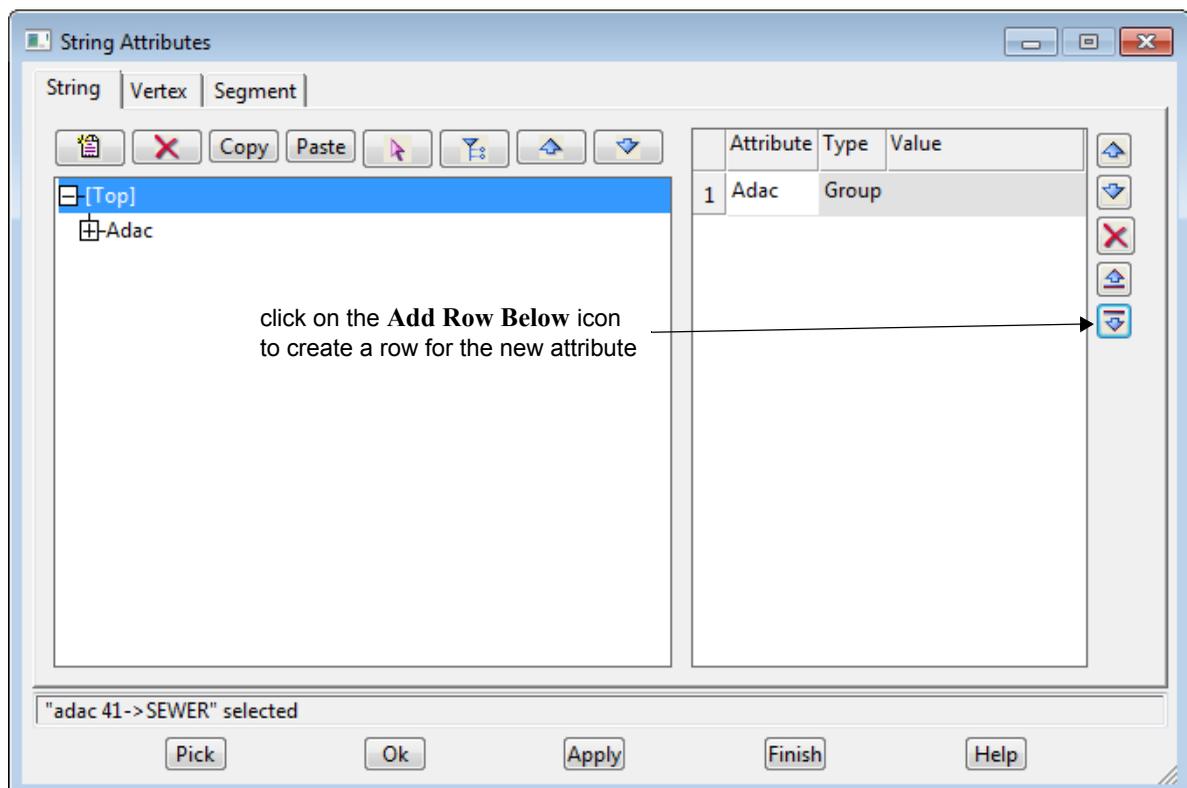
**Note:** To allow the user to identify which *ADAC Asset* string creates a particular row in the *Attributes>Vertex/Pit* grid, the string **name** is copied into the **Comment** column.

For example, if you wanted the *ADAC Attribute* group on the string SEWER to be applied only to vertices of a sewer string with the **pit\_type** attribute equal to **B3**, we would use the **String Attributes** panel to create a text string attribute called **pit\_type** with value **B3**.

To add attributes to the SEWER string, bring up **String Attributes** panel by selecting the option

**Strings =>Properties =>Attributes**

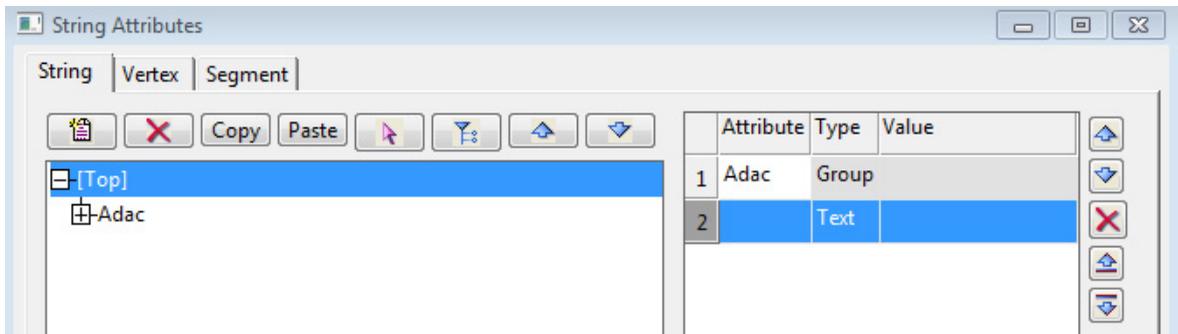
Click on **Pick** and select the string SEWER.



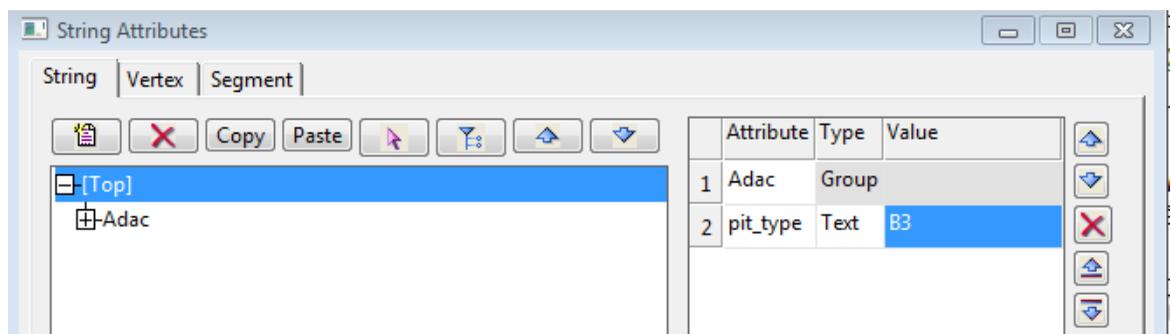
For the **String** tab, highlight the node **[Top]** and then click on the **Add Row Below** icon to create

a new row to use for the attribute **pit\_type**.

**Note:** if there are other first level attributes other than ADAC, highlight delete them.



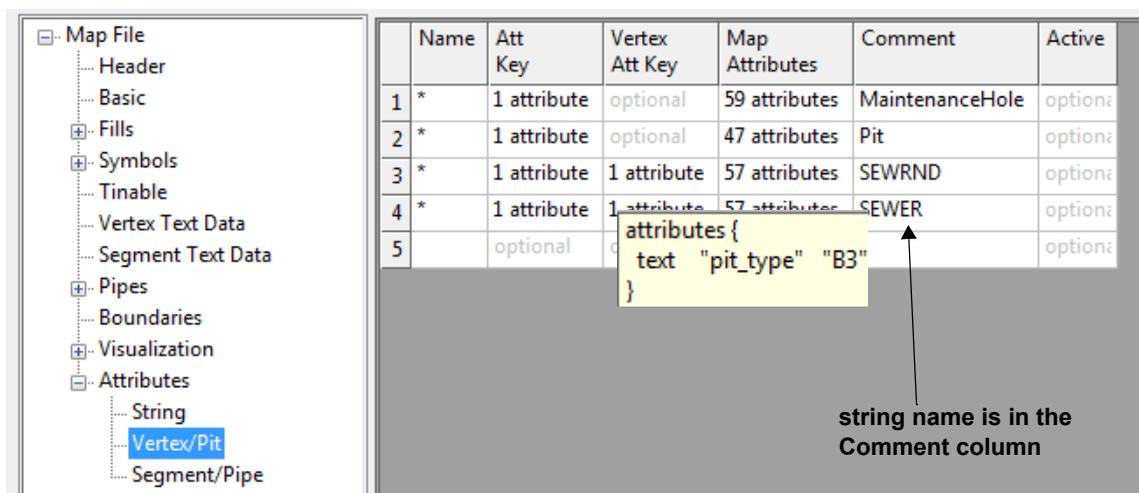
Type in the attribute name **pit\_type** and the value **B3** and then click on the **Apply** button **twice** to add the attribute to the string.



To create the **Map File**, we repeat the process used in [27.6.1.4 Creating the ADAC Attributes for Map Files](#) which is to run the option

**File I/O =>ADAC =>Utilities =>ADAC strings to Map File**

to create the **Map File** and then open the **Map File** and go to the bottom of the **Attributes>Vertex** grid and hover over the **Vertex Att Key** column to see that the attribute **pit\_type** is defined.



Note that the string name SEWER is displayed in the **Comment** column.

So this combination of **Name**, **Att Key** and **Vertex Att Key** in the **Map File** will select the vertex of any sewer string with a text attribute called **pit\_type** with the value **B3**, and give it the ADAC attributes in the **Map Attributes** column.

The ADAC Asset attributes for a **pit** on a **drainage** string are set up the same way except that

the value for **sewer**type is **0** and the ADAC Asset attributes to use are those of a *StormWater>Pits>Pit*.

Continue to the next section [27.6.1.6.3.2 Setting Up the Attributes>Segment Section of the Map File](#) or return to [27.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings](#).

### 27.6.1.6.3.2 Setting Up the Attributes>Segment Section of the Map File

The strings for ADAC Assets that create an entry in the *Attributes>String* section of the **Map File** use the string **name** and **Att Key** as a method of deciding which strings have the **Map Attributes** applied to them.

But the ADAC strings for *Sewerage>PipesNonPressure>PipeNonPressure* or *StormWater>Pipes>Pipe* which create an entry in the *Attributes>Segment/Pipe* section of the **Map File**, already have the **Name** column set to \* and the **Att Key** is used with **sewertype** to select only drainage or sewer strings.

So the only part of the *Attributes>Segment/Pipe* row left help to differentiate between different types of pipes, is the **Segment Att Key**.

Luckily the segments of **12d Model** drainage and sewer string usually have a text **Segment attribute** called **pipe\_type**, or some other segment attributes set during creation, and these can be used to select **Map Attributes** that are more appropriate to that segment.

With that in mind, the **Create Map File from ADAC Data** panel with **Use vertex and segment attributes** ticked **on**, copies the string attribute of the ADAC Asset *Sewerage>PipesNonPressure>PipeNonPressure* or *StormWater>Pipes>Pipe*, to the **Segment Att Key**.

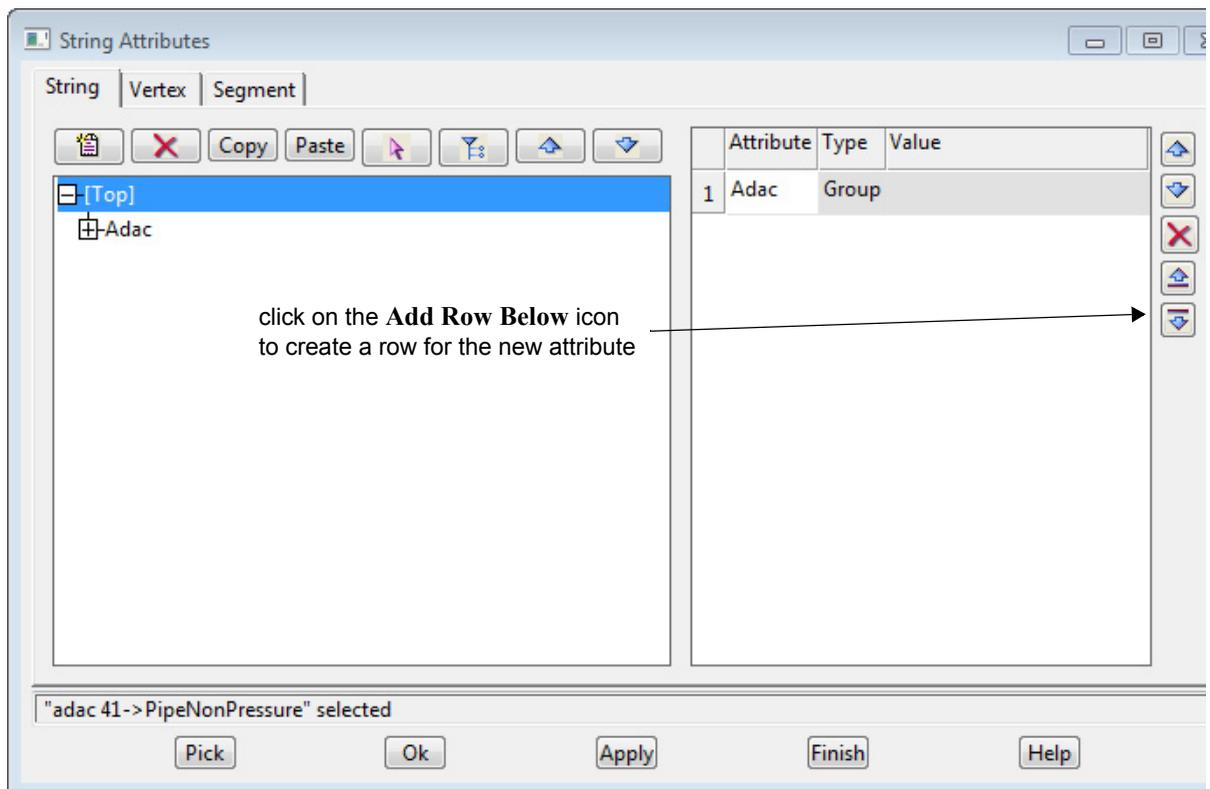
**Note:** To allow the user to identify which *ADAC Asset* string creates a particular row in the *Attributes>Segment/Pipe* grid, the string **name** is copied into the **Comment** column.

For example, if you wanted the *ADAC Attribute* group on the string called **PipeNonPressure** to be applied only to segments of a sewer string with the **pipe\_type** attribute equal to **P3**, we would use the **String Attributes** panel to create a text string attribute called **pipe\_type** with value **P3**.

To add attributes to the **PipeNonPressure** string, bring up **String Attributes** panel by selecting the option

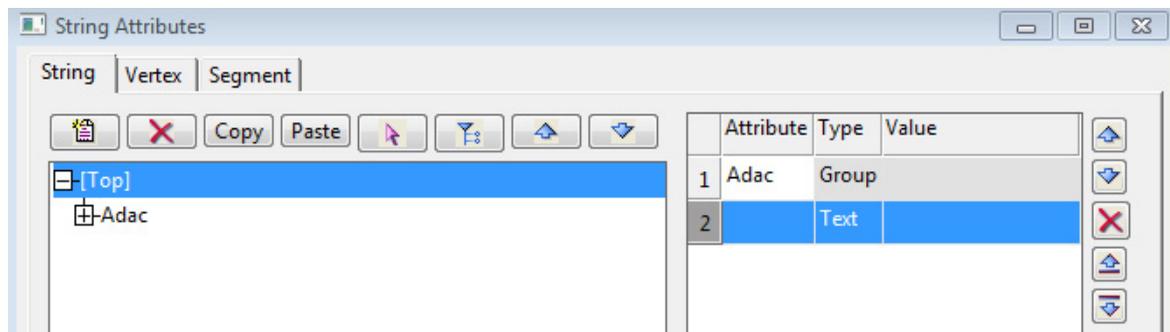
**Strings =>Properties =>Attributes**

Click on **Pick** and select the string **PipeNonPressure**.

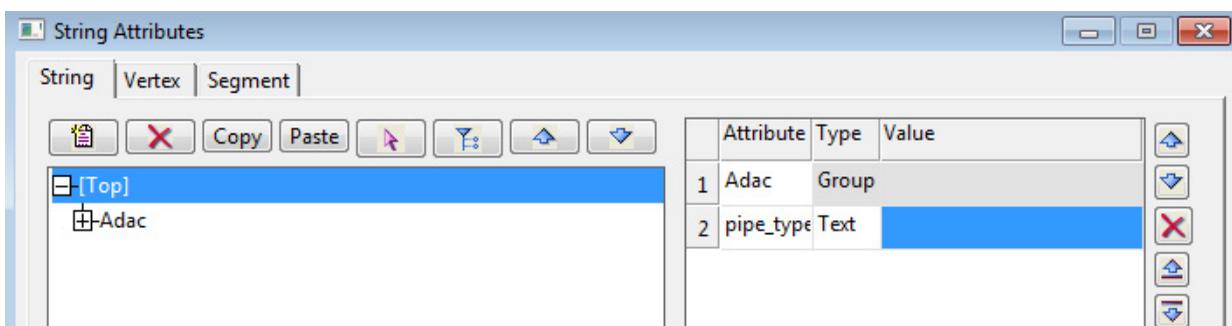


For the **String** tab, highlight the node **[Top]** and then click on the **Add Row Below** icon to create a new row to use for the attribute **pipe\_type**.

**Note:** if there are other first level attributes other than ADAC, highlight delete them.



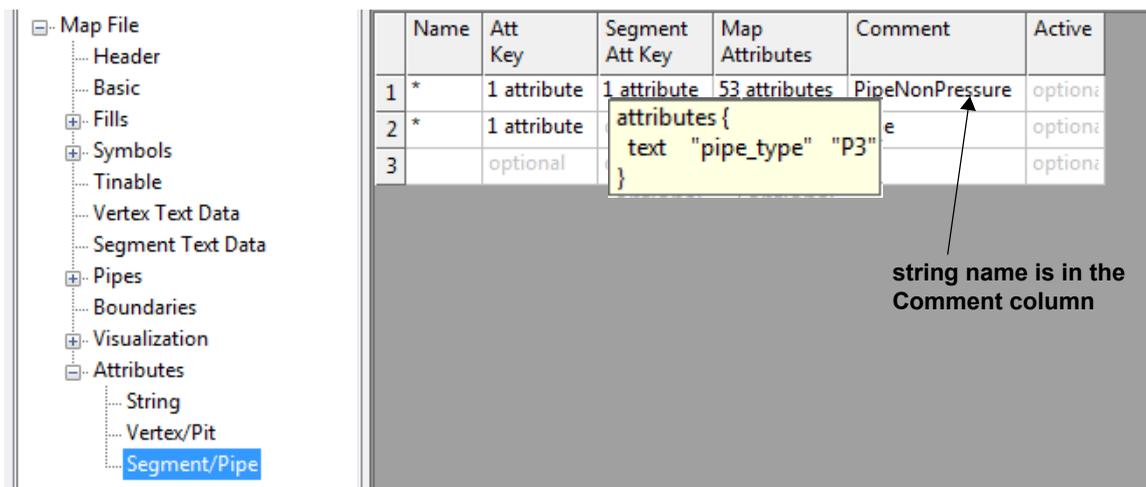
Type in the attribute name **pipe\_type** and the value **P3** and then click on the **Apply** button **twice** to add the attribute to the string.



To create the **Map File**, we repeat the process used in [27.6.1.4 Creating the ADAC Attributes for Map Files](#), which is to run the option

**File I/O =>ADAC =>Utilities =>ADAC strings to Map File**

to create the **Map File** and then open the **Map File** and go to the bottom of the **Attributes>Segment** grid and hover over the **Segment Att Key** column to see that the attribute **pipe\_type** is defined.



Note that the string name **PipeNonPressure** is displayed in the **Comment** column.

So this combination of **Name**, **Att Key** and **Segment Att Key** in the **Map File** will select the segment of any sewer string with a text attribute called **pipe\_type** with the value **P3**, and give it the ADAC attributes in the **Map Attributes** column.

The ADAC Asset attributes for a **pipe** on a **drainage** string are set up the same way except that the value for **sewer**type is **0** and the ADAC Asset attributes to use are those of a *StormWater>Pipes>Pipe*.

Continue to the next section [27.6.1.6.3.3 Bonuses for Drainage and Sewer Strings](#) or return to [27.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings](#).

### 27.6.1.6.3.3 Bonuses for Drainage and Sewer Strings

An ADAC bonus for designers with drainage and sewer strings is that each pit and pipe already has most of the values needed for the ADAC Assets.

For example, a sewer string pit (maintenance hole) knows if the chamber is rectangular or circular, and in both cases, what the dimensions of the chamber and the invert level and the surface level are. The pit name usually contains the MH\_Number and the string name is the LineNumber.

Also from its *pit type*, the Use and most of the other elements of the ADAC Asset *Sewerage>MaintenanceHoles>MaintenanceHole* are known.

Similarly for a sewer string pipe, and the pits and pipes of a drainage string.

And a further bonus is that many of the extra attribute values **do not** have to be the values set for the ADAC Asset set by the **Map File** because later on in the Design chain, the *12dPL ADAC\_Update\_attributes\_from\_drainage\_data\_panel.4do* is run and it updates as many of the ADAC elements for the asset as possible.

For example, it doesn't matter if the **Map File** marked a vertex as having a Circular chamber or a Rectangular chamber because the *12dPL*

*ADAC\_Update\_attributes\_from\_drainage\_data\_panel.4do* will look at the **actual** chamber type of the pit and update the ADAC Asset attributes on the pit with the proper chamber type and size.

For more information on *ADAC\_Update\_attributes\_from\_drainage\_data\_panel.4do*, see [27.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties](#).

Continue to the next section [27.6.1.7 More on Creating the ADAC Map Files](#) or return to [27.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings](#) or [27.6.1.6 Setting Up the Map File Selection Criteria](#).

### 27.6.1.7 More on Creating the ADAC Map Files

In the examples, many of the ADAC Asset values were set by the **Map File**, but others were not and they need to be set in the steps in the *Design* and *Survey* chains that occur after the **Map File** has been applied.

Good procedures and some **data preparation before** running the *Design* or *Survey* chains, will mean that most of the values will be set by the chains and any manual filling in of the missing values by using the **ADAC Editor** will be avoided.

How values such as *Diameter\_mm* are set will vary from Asset to Asset, but they can also vary within the 'same Asset.

For example, for a surveyor picking up *MaintenanceHoles*, they can come prefabricated in standard sizes with known Chamber type and sizes. In fact almost all of the elements for the *MaintenanceHole* would be known - FloorConstruction, FloorMaterial, WallConstruction, WallMaterial, RoofMaterial, LidMaterial. Similarly for Pipes.

So if the surveyor had a special code (string name) for each of the standard *MaintenanceHoles* or *Pipes*, then the ADAC Asset attributes for them can be set up with all the known standard values. Then using the special string name in the **Map File** means that all the values are already set correctly just by applying the **Map File**.

Even if things are not totally standard, some attributes may have a certain value **most of the time**. If that is the case, the "most likely" value is the one that should be used in ADAC Asset in the **Map File** and then only the few that **vary** from the "most likely" value need to be modified at a later stage.

There may also be elements in an ADAC Asset that **the Authority is not interested in** so whatever value they are given by the **Map File** won't have to be changed at any time.

Continue to the next section [27.6.2 What Data Prep is Needed for ADAC](#) or return to [27.6.1 Setting Up Map Files for ADAC](#).

## 27.6.2 What Data Prep is Needed for ADAC

The Data Preparation that is needed on strings before they become ADAC Assets is different for each company. The options that are supplied are on the menu

***File I/O =>ADAC =>User =>Data prep*** See [27.4.9.2 Data Prep](#).

Continue to the next section [27.6.3 Setting up and Running ADAC Chains from the Menus](#) or return to [27.6 Setting Up for ADAC](#).

## 27.6.3 Setting up and Running ADAC Chains from the Menus

The easiest way to run the ADAC *Survey* or *Design* chains with specific pvf files is to run them from a menu. This can be done in two ways:

### (a) Running Specific Map and 12duaf files from Special areas

**12d Model** provides options on the walk right menu **Run 12d ADAC 4.1 Chains** to run ADAC *Survey* or *Design* chains for ADAC 4.1 with specially named Map and 12duaf files in either the working folder for the project (local), or User\_Lib or Library.

See [27.6.3.1 Using the 12d Supplied Menu to Run ADAC Chains](#).

### (b) Running Client Specific Map and 12duaf files

Once you are producing ADAC files for different Clients, you may need to produce different ADAC versions, or need different Map or 12duaf files.

For example you may need to use a different naming convention, or the information required in the ADAC assets is different.

For these situations, you can add your own options to **User Adac** menu.

The user options can run either the ADAC Base Survey or Design chains with specific chain pvf files.

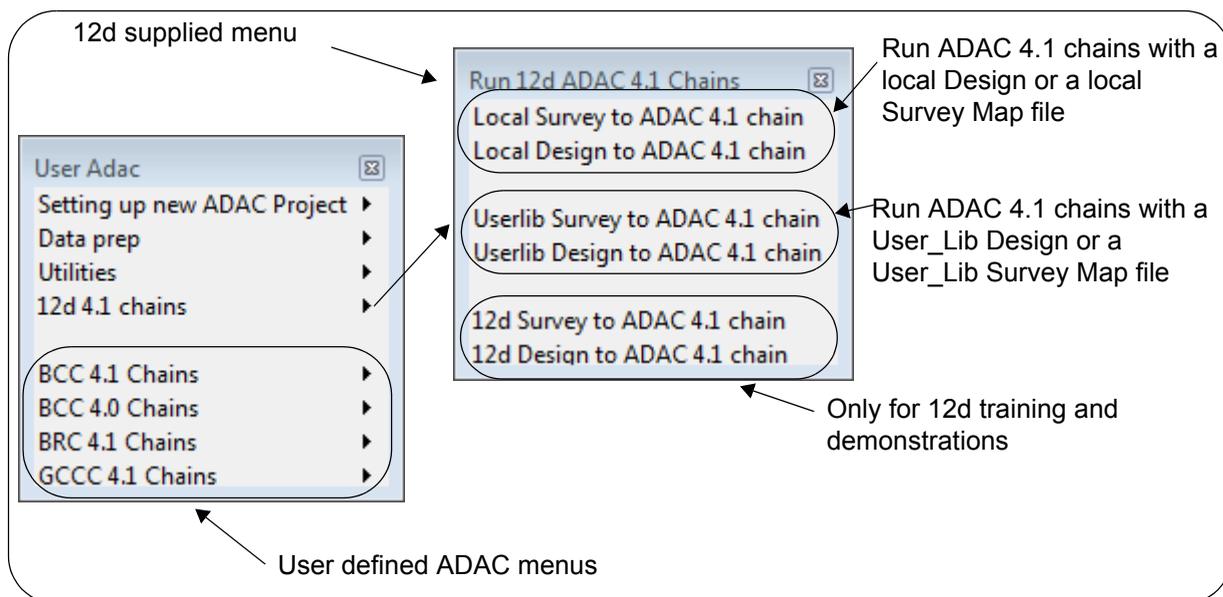
For information on setting up your own options on the **User ADAC** menu, see [27.6.3.2 Setting Up Your User ADAC Menu](#).

Or return to [27.6 Setting Up for ADAC](#).

### 27.6.3.1 Using the 12d Supplied Menu to Run ADAC Chains

**12d Model** provides options on the walk right menu **Run 12d ADAC 4.1 Chains** for the option **12d 4.1 chains** to run the base *ADAC Survey* or *Design* chains for ADAC 4.1 with specially named Map and *12duaf* files in either the **working folder** for the project (**local**) or **User\_Lib**.

These options also use a **tin** called **ground** for surface levels.



- (a) for the working folder for the project (local)

**Local Survey to ADAC 41 chain** runs *ADAC\_Survey\_Base* chain using the files from the **working folder for the project (local)**:

**ADAC\_Local\_Map\_Survey\_to\_ADAC\_41.mapfile**

**Local\_41.12duaf** *this file is optional*

**Local Design to ADAC 41 chain** runs *ADAC\_Design\_Base* chain using the files from the **working folder for the project (local)**:

**ADAC\_Local\_Map\_Design\_to\_ADAC\_41.mapfile**

**Local\_41.12duaf**

Both options use the a **tin** called **ground** to get surface levels.

So to use the **Local** menu items you only need to create the two files and have them in the folder containing the project. This means that they are only available for that project.

- (b) for User\_Lib

**Userlib Survey to ADAC 41 chain** runs *ADAC\_Survey\_Base* chain using the files from the **USER\_LIB** folder:

**ADAC\_Userlib\_Map\_Survey\_to\_ADAC\_41.mapfile**

**Local\_41.12duaf** *this file is optional*

**Userlib Design to ADAC 41 chain** runs *ADAC\_Design\_Base* chain using the files from the **USER\_LIB** folder:

**ADAC\_Userlib\_Map\_Design\_to\_ADAC\_41.mapfile**

**Local\_41.12duaf**

Both options use a **tin** called **ground** to get surface levels.

So to use the **Userlib** menu items, you only need to create the two files and have them in the **User\_Lib** folder. This means they can be used with any project.

Consequently you can use the **Userlib** options for any project and just set up **Local** ones for certain projects.

Continue to the next section [27.6.3.2 Setting Up Your User ADAC Menu](#) or return to [27.6 Setting Up for ADAC](#).

## 27.6.3.2 Setting Up Your User ADAC Menu

To make it easy for your users to run the ADAC chains for either *Survey* or *Design*, or for different versions of ADAC, the easiest way is for the running of the options to be on a menu.

And since what is required could vary from Authority to Authority, and string naming conventions could differ between **12d Model** users, each user needs to set up their own ADAC menus.

The menu **ADAC User** can be modified by users so that is the one we will use.

To allow for all the variations, parametrised base ADAC Design and Survey chains are supplied in the **12d Model** Set\_Ups folder (which most users do not have permission to access) and the variations required for all the different customer cases will be handled by passing parameter values through to these parametrised chains.

Placing the options on the **User ADAC** menu is done by setting up some folders and files in **User** which most users have access to.

To help explain things and the ease of setting up ADAC to work with multiple customers who may vary in what ADAC Assets they require in their ADAC XML files, we will set up a system to satisfy the following scenario:

You have three customers with the abbreviated names **BCC**, **BRC** and **GCCC**.

**BRC** started with ADAC 4.0 but are now moving to ADAC 4.1 and you have jobs in both versions. On some jobs you are providing BRC with Design ADAC XML files and on other jobs you providing BRC with Survey ADAC XML files.

**BCC** are using ADAC 4.1 only and you are providing them with Design and Survey ADAC XML files.

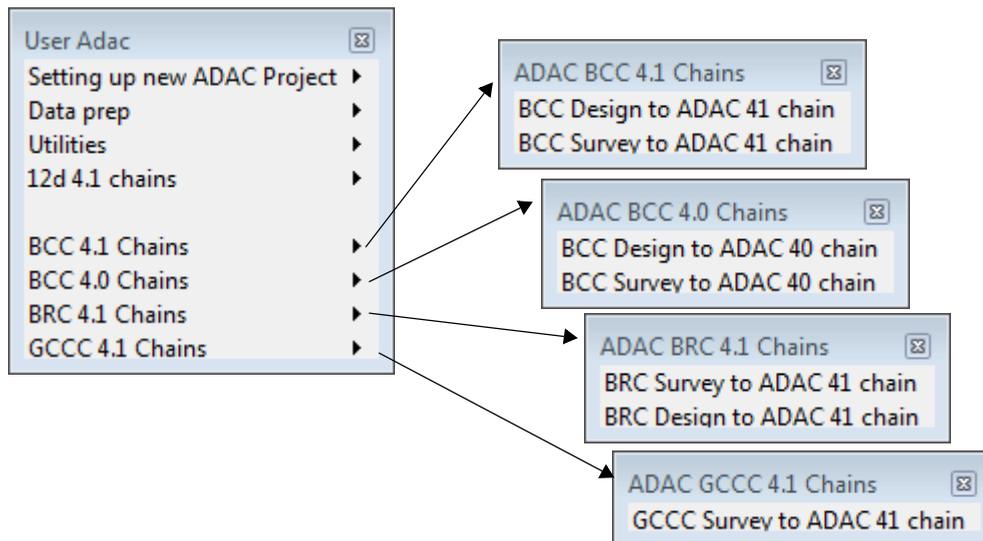
**GCCC** are using ADAC 4.1 only and at this stage and you are only providing them with Survey ADAC XML files.

Due to the nature of design and survey data, the 12d Map Files required for ADAC are always different. But for various reasons, the 12d Map files need to be different for each of the your customers as well.

Of course in real life your customers would never want something totally different but it is good for the scenario.

**Note:** If you are lucky enough that you can use the same configuration for all your customers (or you are an Authority that only has to supply itself) then you only need to set up the one subfolder and it is easiest to use your own abbreviated company name.

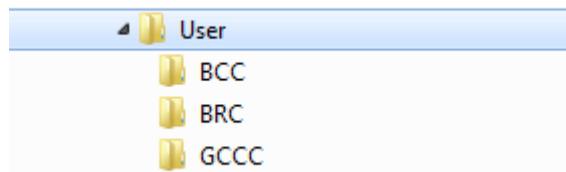
What we want to easily set up is an **User ADAC** menu with walk-rights:



The steps to achieve this are:

- (a) In the file **usermenu.4d** (which is usually in **User**), add the following line at the top of the file.
 

```
#include "ADAC_customer_user_menu.4d"
```
- (b) Create subfolders of **User** with the abbreviated name for each of your customers.



- (c) Inside the subfolder (for example, inside **UserBCC**) there will be an appropriate **ADAC pvf** file

The ADAC **pvf** file is a **parameter value file** which is used in to pass parameters to both the **ADAC Design chain** and **ADAC Survey chain**.

In the ADAC **pvf** files themselves, there is five parameters that you have to set the values for:

- (i) For the text parameter **company**, the Value which is the abbreviated company name. For example, **BCC**.
- (ii) For the text parameter **adac\_version\_by\_ten**, the Value **41** if you are generating ADAC 4.1.0 files, or **40** if you are generating ADAC 4.0 files.
- (iii) **user\_attributes\_conversion\_type**
- (iv) For the text parameter **survey\_finished\_tin**, the name of a tin which is used to get some z-values for some tins in some macros.
- (v) For the text parameter **design\_finished\_tin**. The name of a tin which may be used to get some z-values for some Assets in some macros

There are four other parameters in the **pvf** files, **survey\_map\_file**, **design\_map\_file**, **user\_adac\_attribute\_file** and **adac\_12d\_attribute\_file** but they are fully defined once company and **adac\_version\_by\_ten** have values.

- (vi) **survey\_map\_file** is

- \$USER\_LIB\ADAC\_[company]\_Map\_Survey\_to\_ADAC\_[adac\_version\_by\_ten].mapfile
- (vii) design\_map\_file is  
\$USER\_LIB\ADAC\_[company]\_Map\_Design\_to\_ADAC\_[adac\_version\_by\_ten].mapfile
- (viii) user\_adac\_attribute\_file is  
\$USER\_LIB\[company]\_[adac\_version\_by\_ten].12duaf
- (ix) adac\_12d\_attribute\_file is  
\$LIB\adac\_12d\_[adac\_version\_by\_ten].12duaf

To easily tell the different **pvf** files apart when you have them for different companies and ADAC versions, the **abbreviated company name** and the **adac\_version\_by\_ten** are used in the name of the **pvf** file.

So for **company** with value **BCC** and **adac\_version\_by\_ten** with value **41**, the **pvf** file is called

**ADAC\_BCC\_41.pvf**

And this **pvf** file must have the following contents:

```
<Chain_Parameters>
<Text_Parameter>
  <Name>"company"</Name>
  <Comment>""</Comment>
  <Value>"BCC"</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"design_map_file"</Name>
  <Comment>""</Comment>
  <Value>"$USER_LIB\ADAC_[company]_Map_Design_to_ADAC_[adac_version_by_ten].mapfile"</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"survey_map_file"</Name>
  <Comment>""</Comment>
  <Value>"$USER_LIB\ADAC_[company]_Map_Survey_to_ADAC_[adac_version_by_ten].mapfile"</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"user_attributes_conversion_type"</Name>
  <Comment>""</Comment>
  <Value>"BCC"</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"design_finished_tin"</Name>
  <Comment>"ground"</Comment>
  <Value>""</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"survey_finished_tin"</Name>
  <Comment>""</Comment>
  <Value>"ground"</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"adac_version_by_ten"</Name>
  <Comment>""</Comment>
  <Value>"41"</Value>
</Text_Parameter>
<Text_Parameter>
  <Name>"user_adac_attribute_file"</Name>
  <Comment>""</Comment>
  <Value>"$USER_LIB\[company]_[adac_version_by_ten].12duaf"</Value>
</Text_Parameter>
```

```

<Text_Parameter>
  <Name>"adac_12d_attribute_file"</Name>
  <Comment>""</Comment>
  <Value>"$LIB\adac_12d_[adac_version_by_ten].12duaf"</Value>
</Text_Parameter>
</Chain_Parameters>

```

The different **pvf** files can be created by copying the **ADAC\_BCC\_41.pvf** file and changing the **BCC** and **41** to what is needed. This can be done using a text editor, or the **12d Model** options

*Utilities => Chains => Parameters => Create or Copy or Edit.*

So in the folder **User\BCC**, you need two pvf files - **ADAC\_BCC\_41.pvf** and **ADAC\_BCC\_40.pvf**

In the folder **User\BRC**, you need the one pvf file - **ADAC\_BRC\_41.pvf**

And in the folder **User\GCCC**, you need the one pvf file - **ADAC\_GCCC\_41.pvf**.

- (d) Inside each customer subfolder (for example, inside **User\BCC**) there must be a file called **ADAC\_customer\_user\_menu.4d** which creates the walk right menus for that customer that go on your **User** menu.

In our example, **BCC** requires both **ADAC 4.1** and **ADAC 4.0** so two **walk** right menus are needed.

So in the folder **User\BCC**, the contents for

**ADAC\_customer\_user\_menu.4d**

are:

```

Button "BCC 4.1 Chains" {
  Walk_Right "ADAC BCC 4.1 Chains"
}
Button "BCC 4.0 Chains" {
  Walk_Right "ADAC BCC 4.0 Chains"
}

```

the text on the menu will be **BCC 4.1 Chains**

and it has a walk right menu and the definition for the walk right menu is in a menu called **ADAC BCC 4.1 Chains**

**BRC** only required ADAC 4.1 so only one walk right menu is required. and **ADAC\_customer\_user\_menu.4d** in the folder **User\BRC** is

```

Button "BRC 4.1 Chains" {
  Walk_Right "ADAC BRC 4.1 Chains"
}

```

the text on the menu will be **BRC 4.1 Chains**

and it has a walk right menu and the definition for the walk right menu is in a Menu called **ADAC BRC 4.1 Chains**

In the folder **User\GCCC**, the **ADAC\_customer\_user\_menu.4d** file is almost identical to that for BRC that BRC is replaced by GCCC.

So now we need to define walk-right menus mentioned in the **ADAC\_customer\_user\_menu.4d** files.

- (e) Inside each customer folder, the walk right menus are defined in a file called **ADAC\_customer\_walk\_right\_menu.4d**

The walk right menus contain the options that run the required ADAC Design and ADAC Survey chains.

In our example, **BCC** has two walk right menus and they both need options to run Design and Survey chains.

So in the folder **User\BCC**, the contents of **ADAC\_customer\_walk\_right\_menu.4d** are

```
Menu "ADAC BCC 4.1 Chains" {
  Button "BCC Design to ADAC 41 chain" {
    Command "chain -pvf $USER/BCC/ADAC_BCC_41.pvf $LIB/ADAC_design_base.chain"
  }
  Button "BCC Survey to ADAC 41 chain" {
    Command "chain -pvf $USER/BCC/ADAC_BCC_41.pvf $LIB/ADAC_survey_base.chain"
  }
}

Menu "ADAC BCC 4.0 Chains" {
  Button "BCC Design to ADAC 40 chain" {
    Command "chain -pvf $USER/BCC/ADAC_BCC_40.pvf $LIB/ADAC_design_base.chain"
  }
  Button "BCC Survey to ADAC 40 chain" {
    Command "chain -pvf $USER/BCC/ADAC_BCC_40.pvf $LIB/ADAC_survey_base.chain"
  }
}
```

**ADAC design chain**

**pvf file for BCC and ADAC 4.1**

**ADAC survey chain**

**BRC** only require ADAC 4.1 but do need both Design and Survey so only one walk right menu is required but it needs an option for Design and an option for Survey on it. So the **ADAC\_customer\_walk\_right\_menu.4d** in the folder **User\BRC** is

```
Menu "ADAC BRC 4.1 Chains" {
  Button "BRC Design to ADAC 41 chain" {
    Command "chain -pvf $USER/BRC/ADAC_BRC_41.pvf $LIB/ADAC_design_base.chain"
  }
  Button "BRC Survey to ADAC 41 chain" {
    Command "chain -pvf $USER/BRC/ADAC_BRC_41.pvf $LIB/ADAC_survey_base.chain"
  }
}
```

**ADAC design chain**

**pvf file for BRC and ADAC 4.1**

**ADAC survey chain**

**GCCC** only require ADAC 4.1 and only need Survey so only one walk right menu is required with just the one option for Survey on it. So the **ADAC\_customer\_walk\_right\_menu.4d** in the folder **User\GCCC** is

```

Menu "ADAC GCCC 4.1 Chains" {
  Button "GCCC Survey to ADAC 41 chain" {
    Command "chain -pvf($USER/GCCC/ADAC_BRC_41.pvf) $LIB/ADAC_survey_base.chain"
  }
}

```

*pvf file for GCCC and ADAC 4.1*
*ADAC survey chain*

- (f) Finally the **ADAC\_customer\_user\_menu.4d**'s for each of the customers needs to be included in the **ADAC User** menu

This is done by having a file in the **User** folder called **ADAC\_customer\_user\_menu.4d**

(Aside: Yes this is the same name as in the customer folders. If it causes confusion then we will change it. Maybe it should have an s in it **ADAC\_customers\_user\_menu.4d**)

This is the file that was included in the **User** menu by the  
`#include "ADAC_customer_usr_menu.4d" statement.`

This file first includes all the walk right menus for the different customers on the **User ADAC** menu, and then includes the definitions of the walk right menus.

```

Menu "User Adac" {
  #include "BCC\ADAC_customer_user_menu.4d"
  #include "BRC\ADAC_customer_user_menu.4d"
  #include "GCCC\ADAC_customer_user_menu.4d"
}

#include "BCC\ADAC_customer_walkright_menu.4d"
#include "BRC\ADAC_customer_walkright_menu.4d"
#include "GCCC\ADAC_customer_walkright_menu.4d"

```

*places each walk right menu on the User ADAC menu*

*defines the walk right menus*

Continue to the next section [27.6.4 Setting Up ADAC Templates](#), or return to [27.6.3 Setting up and Running ADAC Chains from the Menus](#) or [27.6 Setting Up for ADAC](#).

## 27.6.4 Setting Up ADAC Templates

When creating *ADAC Header* using the **ADAC =>Create Header**, or *ADAC Assets* using the **ADAC =>Create Asset** option, Templates can be used to automatically fill in many of the values of ADAC elements.

See

[27.6.4.1 ADAC Header Templates](#)

[27.6.4.2 ADAC Asset Templates](#)

### 27.6.4.1 ADAC Header Templates

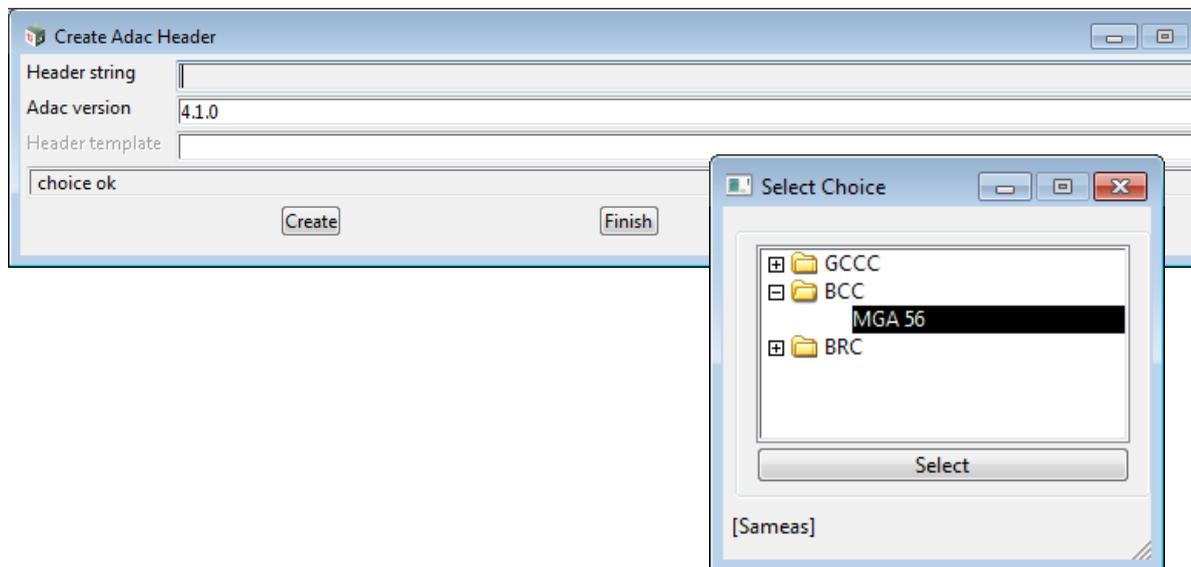
In most firms, many of the values in an *ADAC Header* would be the same for each new ADAC job.

For example, all your jobs could be in *MGA zone 56*, using the *Horizontal Datum GDA 94* and *Vertical Datum AHD*. Or you always have the three surveyor names in the *ADAC Header*.

To prevent having to reenter this data every time you create a new ADAC Header, users can create *ADAC Header Templates* with information for any new ADAC Header already filled in.

When you have *ADAC Header Templates*, the **Template** field pop up in the **Create ADAC Header** panel lists all your *ADAC Header Templates* for the selected *ADAC version*, as a folder structure.

When the new *ADAC Header* is created, it will have **all** the values from the selected *Header Template*.

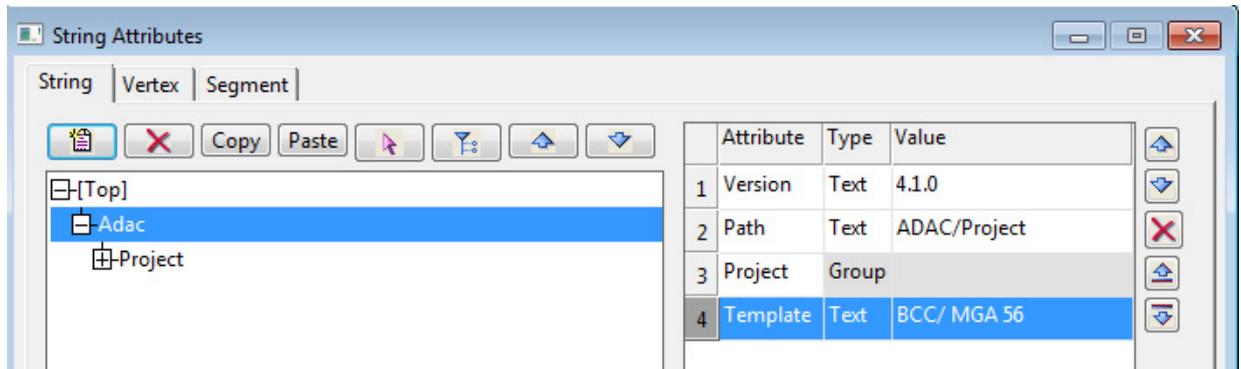


So for the new *ADAC Header*, only those values not covered by the *Header Template* need to be entered.

To create an *ADAC Header Template*, you

1. create an *ADAC Header* for the required *ADAC version*, and with the *ADAC values* that you want the *Header Template* to have
2. place it in the model *ADAC Header Templates*
3. in the string attributes, add a **Text** attribute called **Template** into the **top** level of the *ADAC attribute group*, with its value being the required folder structure for displaying the *Header Template* with the names for each folder level being separated by a forward slash (/).

For example, to have a **Header Template** called **MGA 96** and have it display in a folder called **BCC**, the value for the attribute **Template** is **BCC/MGA 96**



To use the **Header Templates** in another project you can either:

- (a) write the model **ADAC Header Templates** out as a 12da file and read it into new ADAC projects.
- (b) have the model **ADAC Header Templates** in a project and share that project into your new ADAC projects.
- (c) write the models **ADAC Header Templates** and **ADAC Asset Templates** out to the 12da file **ADAC\_Templates.4da** and place the file in User\_Lib.

The option:

ADAC =>User =>Setting up a new ADAC Project =>Read in templates from User\_Lib  
 can then be used in any project to read *User\_Lib\ADAC\_Templates.4da* in, and hence the models **ADAC Header Templates** and **ADAC Asset Templates** into the new project.

See [27.4.9.1.3 Reading in Templates from User\\_Lib](#).

Continue to the next section [27.6.4.2 ADAC Asset Templates](#) or return to [27.6.4 Setting Up ADAC Templates](#).

## 27.6.4.2 ADAC Asset Templates

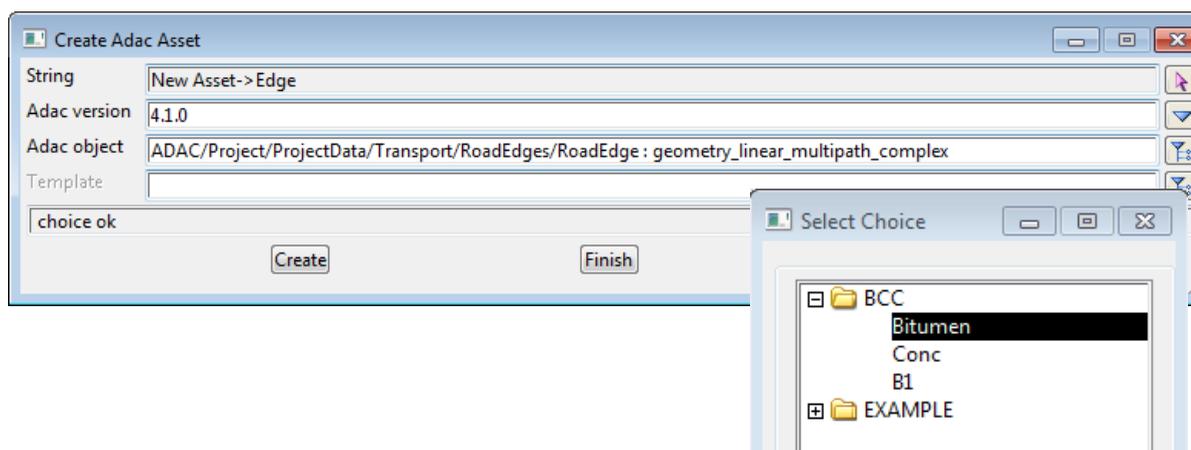
When creating *ADAC Assets* by hand, there would be some the values in an *ADAC Asset* that would be the same each time the *ADAC Asset* was use in ADAC job.

For example, a *Transport>RoadEdges>RoadEdge* with **Type *Bitumen*** and **Owner *Council*** may be regularly required.

To prevent having to reenter this data every time you create a new *ADAC Asset*, users can create *ADAC Asset Templates* with information for any new *ADAC Asset* already filled in.

When you have *ADAC Asset Templates*, the **Template** field pop up in the **Create ADAC Asset** panel, lists all your *ADAC Asset Templates* for the selected *ADAC version* and the selected *ADAC Asset*, in a folder structure.

When the new *ADAC Asset* is created, it will have **all** the values from the selected *Asset Template*.

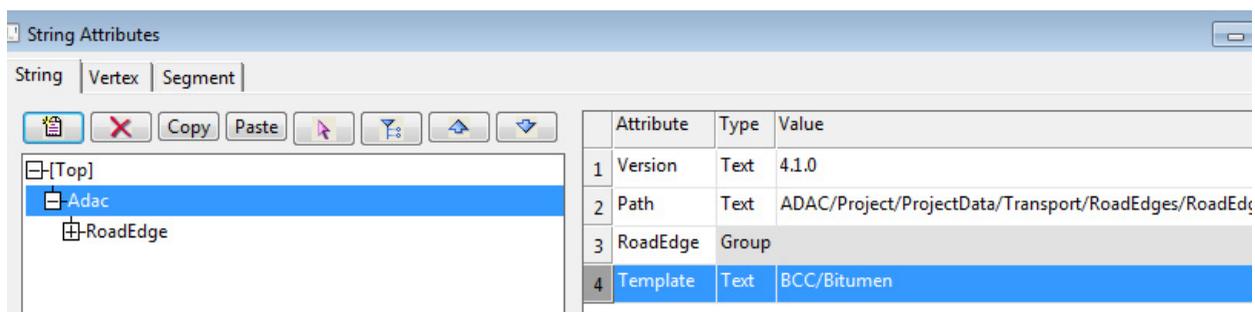


So for the new *ADAC Asset*, only those values not covered by the *Asset Template* need to be entered.

To create an *ADAC Asset Template*, you

1. create an *ADAC Asset* for the required *ADAC version*, and with the *ADAC values* that you want the *Asset Template* to have
2. place it in the model *ADAC Asset Templates*
3. in the string attributes, add a **Text** attribute called **Template** into the **top** level of the *ADAC attribute group*, with its value being the required folder structure for displaying the *Asset Template* with the names for each folder level being separated by a forward slash (/).

For example, to have an *Asset Template for a Road Edge* called **Bitumen**, and have it display in a folder called **BCC**, the value for the attribute **Template** is **BCC/Bitumen**



To use the **Asset Templates** in another project you can either:

- (a) write the model **ADAC Asset Templates** out as a 12da file and read it into new ADAC projects.
- (b) have the model **ADAC Asset Templates** in a project and share that project into your new ADAC projects.
- (c) write the models **ADAC Header Templates** and **ADAC Asset Templates** out to the 12da file **ADAC\_Templates.4da** and place the file in *User\_Lib*.

The option:

**ADAC =>User =>Setting up a new ADAC Project =>Read in templates from User\_Lib**

can then be used in any project to read *User\_lib\ADAC\_Templates.4da* in, and hence the models **ADAC Header Templates** and **ADAC Asset Templates** into the new project.

See [27.4.9.1.3 Reading in Templates from User\\_Lib](#).

Continue to the next section [27.6.5 Setting Up Your User Keys](#) or return to [27.6 Setting Up for ADAC](#).

## 27.6.5 Setting Up Your User Keys

Two things you will do regularly when doing ADAC work is to bring up the ADAC menu and to bring up the **String Attributes** panel.

So in the standard **12d Model 11** installation, the function keys are assigned so that:

Pressing **F12** brings up the **ADAC** menu.

Pressing **F8** brings up the **Strings Attributes** panel.

If you wish to change the use of function keys then you can define the way you want the *function keys* to work.

For example, you may want

Press **F11** to bring up the **ADAC** menu.

Press **F12** to bring up the **String Attributes** panel.

In **12d Model** the actions of the function keys is controlled by *userkeys.4d*. There may already be a copy in your **User** folder. If not, copy it over from the folder in **Program Files**:

```
12d\12dmodel\11.00\set_ups.
```

Then just replace the existing lines for defining **f11** and **f12** with (lines preceded by *//* are comment lines).

```
// mod for f11
```

```
f11    panel "ADAC"
```

```
// mod for f12
```

```
f12    panel "String Attributes"
```

Return to [27.6 Setting Up for ADAC](#).

# 28 12d XML File Format

**Extensible Markup Language (XML)** is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the **World Wide Web Consortium's (W3C)** XML Specifications which are free open standards.[

The **12d XML** file format is a text file definition from **12d Solutions** which is used for reading and writing out string data from **12d Model**. 12d XML files normally end in **.12dxml**.

The **12d XML** file is a **Unicode** file.

This document is for the **12d XML** file format used in **12d Model Version 11**.

For general comments see:

[28.1 General Information about XML](#)

[28.2 General Information about a 12d XML File](#)

For the 12d XML definitions see:

[28.4 Attributes](#)

[28.5 Model](#)

[28.6 Elements Contained in Models](#) which includes

[28.6.1 Tin](#)

[28.6.2 Super Tin](#)

[28.6.5 Arc String](#)

[28.6.6 Circle String](#)

[28.6.7 Drainage String](#)

[28.6.8 Feature String](#)

[28.6.9 Plot Frame String](#)

[28.6.10 Super String](#)

[28.6.11 Super Alignment String](#)

[28.6.12 Text String](#)

[28.6.13 Trimesh](#)

For documentation on the **12d Archive (12da)** file format, see [36 12d Archive File Format](#).

## 28.1 General Information about XML

### (Unicode) Character

By definition, an XML document is a string of characters. Almost every legal Unicode character may appear in an XML document.

### Markup and Content

The characters making up an XML document are divided into *markup* and *content*, which may be distinguished by the application of simple syntactic rules.

Generally, strings that constitute markup either begin with the character `<` and end with a `>`, or they begin with the character `&` and end with a `;`.

Strings of characters that are not markup are content.

However, in a CDATA section, the delimiters `<![CDATA[` and `]]>` are classified as markup, while the text between them is classified as content. In addition, whitespace before and after the outermost element is classified as markup.

### Characters "`<`", "`>`" and "`&`"

The characters "`<`", "`>`" and "`&`" are key syntax markers and may never appear in content outside a CDATA section. They need to be represented by special escape sequences:

`&lt;` represents "`<`"

`&gt;` represents "`>`"

`&amp;` represents "`&`"

### Tag

An XML tag is a markup construct that begins with `<` and ends with `>`.

Tags come in three flavours:

- (a) start-tags - for example: `<section>`
- (b) end-tags - for example: `</section>`
- (c) empty-element tags - for example: `<line-break />`

### XML Element

A logical document component which either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag.

The characters between the start- and end-tags, if any, are the element's content, and may contain markup, including other elements, which are called child elements.

An example of an element is `<Greeting>Hello, world.</Greeting>`.

Another is `<line-break />`.

**Note:** Because elements are *12d Model* items that are in a model, in the documentation of 12d XML we will refrain from using element for the element in XML. Instead we will use the words *keyword block* to refer to special XML Elements in 12d XML.

### Empty XML Elements `<keyword/>`

When an XML element has no content it is called an **empty** element.

For example `<name> </name>`

There is special shorthand for empty elements:

`<keyword/>` is shorthand for `<keyword></keyword>`

### XML Attribute

A markup construct consisting of a name/value pair that exists within a start-tag or empty-element tag. In the example (below) the element **img** has two attributes, **src** and **alt**:

```

```

Another example would be

```
<step number="3">Connect A to B.</step>
```

where the name of the attribute is "number" and the value is "3".

An XML attribute can only have a single value and each attribute can appear at most once on each element.

**Note:** Because attributes are fundamental **12d Model** items, in the documentation of 12d XML the word attribute will refer to **12d Model** attributes.

The words **XML attribute** will always be used when there is need to refer to an XML attribute.

### XML declaration

XML documents may begin by declaring some information about themselves, as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

### Escaping

XML provides escape facilities for including characters which are problematic to include directly. For example:

There are five predefined entities:

**&lt;** represents "<"

**&gt;** represents ">"

**&amp;** represents "&"

**&apos;** represents "'

**&quot;** represents "

**&#xa;** represents a new line.

### XML Comments

Comments may appear anywhere in a document outside other markup. Comments cannot appear before the XML declaration.

Comments start with "**<!--**" and end with "**-->**".

For compatibility with SGML, the string "--" (double-hyphen) is not allowed inside comments; this means comments cannot be nested.

The ampersand has no special significance within comments, so entity and character references are not recognized as such, and there is no way to represent characters outside the character set of the document encoding.

An example of a valid comment: "**<!--no need to escape <code> & such in comments-->**"

Continue to the next section [28.2 General Information about a 12d XML File](#) or return to [28 12d XML File Format](#).

## 28.2 General Information about a 12d XML File

### Unicode

*12d XML* file is a Unicode file.

### Blank lines

Unless they are part of a string of characters making up text, blank lines are ignored.

### Names of models, tins, styles, colours and attributes

Models, tins, styles (linestyles), colours and attributes can include the characters a to z, A to Z, 0 to 9 (alphanumeric characters) and space. Leading and trailing spaces are ignored. The names can be up to 255 characters in length.

The names for models, tins, styles, colours or attributes can not be blank.

The names for models, tins, styles and colours can contain upper and lower alpha characters which are stored, but the set of model names, tin names, style names, colour names or attribute names for an object *must be unique when case is ignored*. For example, the model name "Fred" will be stored as "Fred" but "FRED" is considered to be the *same* model name as "Fred".

### String names

String names can include the characters a to z, A to Z, 0 to 9 (alphanumeric characters), space, decimal point (.), plus (+), minus (-), comma (,), open and closed round brackets and equals (=).

Leading and trailing spaces are ignored.

String names do not have to be unique and can be blank.

String names can contain upper and lower alpha characters which are retained but case is ignored when selecting by string name. That is, the string name **Fred** will be stored as **Fred** but **FRED** is not considered to be a different string name.

### Keywords Blocks

There are many regularly used blocks of information in 12d XML that are identified and documented by keywords.

The keyword and its block consist of a starting **<keyword>**, followed by the information in the keyword block, and ending in **</keyword>**

That is

**<keyword>** information in the keyword block **</keyword>**

Continue to the next section [28.3 Regularly Used Keyword Blocks](#) or return to [28 12d XML File Format](#).

## 28.3 Regularly Used Keyword Blocks

In the documentation of **12d XML** the term **keyword block** refers to a `<keyword>` followed by various information then a `</keyword>`.

For the definition of some of the regularly used keyword blocks used in the 12d XML see:

[28.3.1 Name](#)

[28.3.2 Colour](#)

[28.3.3 Line Style](#)

[28.3.4 Chainage](#)

[28.3.5 Weight](#)

[28.3.6 Interval](#)

[28.3.7 Time Created](#)

[28.3.8 Time Updated](#)

[28.3.9 Breakline](#)

[28.3.10 Null](#)

[28.3.9 Breakline](#)

Or return to [28 12d XML File Format](#).

### 28.3.1 Name

The format of the **name** keyword block is:

```
<name>name_text</name>
```

where **name\_text** is a string of characters.

What characters can be in the name depends on where the name is used. See [Names of models, tins, styles, colours and attributes](#) and [String names](#).

Continue to the next section [28.3.2 Colour](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

### 28.3.2 Colour

The format of the **colour** keyword block is:

```
<colour>colour_name</colour>
```

where **colour\_name** is a string of characters that is to be the name of a colour or the colour number.

When reading a 12d XML file, there is a **current colour**, which has the default value of **red**, and when a **colour** command is read, the **current colour** is set to **colour\_name**.

When strings are read in a 12d XML file, they are given the *current colour*.

This can be overridden for a string by a *string colour command* inside the string command defining that string. For the definition of the string commands, see [28.6.3 String Header Block](#).

Continue to the next section [28.3.3 Line Style](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

### 28.3.3 Line Style

The format of the **line style** keyword block is:

```
<style>line_style_name</style>
```

where *line\_style\_name* is the name of a line style. It is a string of characters.

When reading a 12d XML file, there is a **current linestyle**, which has the default value of **1**, and when a **style** command is read, the **current linestyle** is set to *linestyle\_name*.

When strings are read in a 12d XML file, they are given the *current linestyle*.

This can be overridden for a string by a *string style command* inside the string command defining that string. For the definition of the string command, see [28.6.3 String Header Block](#).

Continue to the next section [28.3.4 Chainage](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

### 28.3.4 Chainage

The format of the **chainage** keyword block is:

```
<chainage> chainage_real </chainage>
```

where *chainage\_real* is a real value.

Continue to the next section [28.3.5 Weight](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

### 28.3.5 Weight

The format of the **weight** keyword block is:

```
<weight> weight_real </weight>
```

where *weight\_real* is a real value.

Continue to the next section [28.3.6 Interval](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

### 28.3.6 Interval

For all elements other than the super string, the format of the **interval** keyword block is:

```
<interval> interval_real </interval>
```

where *interval\_real* is a real value.

For a super string, the format of the **interval** keyword block is:

```
<interval>
  <chord_arc> chord_arc_real</chord_arc>
  <distance> distance_real</chord_arc>
</interval>
```

where *chord\_arc\_real* and *distance\_real* are real values.

Continue to the next section [28.3.7 Time Created](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

## 28.3.7 Time Created

The format of the **time\_created** keyword block is:

```
<time_created>time_text</time_created>
```

where **time\_text** is a string of characters in the W3C time format.

*DD-MMM-YYYYThh:mm:ssZ*

and

*dd* in the day of the month

*MMM* in the first three letters of the month

*YYYY* in the year

*hh* in the hour in the 24-hour clock

*mm* in the number of minutes

*ss* in the number of seconds

For example, 28-Apr-2015T06:42:45Z

Continue to the next section [28.3.8 Time Updated](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

## 28.3.8 Time Updated

The format of the **time\_updated** keyword block is:

```
<time_updated>time_text</time_updated>
```

where **time\_text** is a string of characters in the W3C time format.

*DD-MMM-YYYYThh:mm:ssZ*

and

*dd* in the day of the month

*MMM* in the first three letters of the month

*YYYY* in the year

*hh* in the hour in the 24-hour clock

*mm* in the number of minutes

*ss* in the number of seconds

For example, 28-Apr-2015T06:42:45Z

Continue to the next section [28.3.9 Breakline](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

## 28.3.9 Breakline

The format of the **breakline** keyword block is:

```
<breakline> breakline_type_text </breakline>
```

where **breakline\_type\_text** is text and can only have the values **point** or **line**.

When reading a 12d XML file, there is a **current breakline type**, which has the default value of **point**, and when a **breakline** command is read, the **current breakline type** is set to **breakline\_type\_text**.

When strings are read in a 12d XML file, they are given the *current breakline type*.

This can be overridden for a string by a *string breakline command* inside the string command defining that string. For the definition of the string command, see [28.6.3 String Header Block](#).

Continue to the next section [28.3.10 Null](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28.12d XML File Format](#).

## 28.3.10 Null

NOT CERTAIN ABOUT NULL ??

The format of the **null** command is:

```
null null_value
```

When reading a 12d XML file, there is a **current null value**, which has the default value of **-999**, and when a **null** command is read, the **current null value** is set to *null\_value*.

When strings are read in a 12d XML file and the string has z-values equal to *null\_value*, then the z-value is replaced by the **12d Model** null value.

This can be overridden for a string by a *null\_value command* inside the string command defining that string. For the definition of the string command, see [28.6.3 String Header Block](#).

Continue to the next section [28.3.11 Radius](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28.12d XML File Format](#).

Continue to the next section [28.4 Attributes](#) or return to [28.12d XML File Format](#).

## 28.3.11 Radius

The format of the **radius** keyword block is:

```
<radius> radius_real </radius>
```

where *radius\_real* is a real value.

Continue to the next section [28.3.12 data\\_2d](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28.12d XML File Format](#).

## 28.3.12 data\_2d

For some strings, there is a constant z for the entire string, or even no z value at all. For such strings only the (x,y) coordinates are required for each vertex and no space is taken up by redundant z values. Vertex data with no z-values is written out in a **data\_2d** block.

The definition of a **data\_2d** block is:

```
<data_2d>
  <p>x_value_1 y_value_1</p>
  <p>x_value_2 y_value_2</p>
  ...
  <p>x_value_n y_value_n</p>
</data_2d>
```

where (x\_value\_i, y\_value\_i) are the 2D coordinates of the i'th vertex.

Continue to the next section [28.3.13 data\\_3d](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28.12d XML File Format](#).

### 28.3.13 data\_3d

For most string, the z value can vary for each vertex along the string and so the (x,y,z) values are required for each vertex. This vertex data is written out as a **data\_3d** block.

The definition of a **data\_3d** block is:

```
<data_3d>
  <p>x_value_1  y_value_1  z_value_1</p>
  <p>x_value_2  y_value_2  z_value_2</p>
  ...
  <p>x_value_n  y_value_n  z_value_n</p>
</data_3d>
```

where (x\_value\_i, y\_value\_i, z\_value\_i) are the 3D coordinates of the i'th vertex.

For example, for a string of 5 vertices

```
<data_3d>
  <p>42578.27649249  37366.79821468  null</p>
  <p>42523.36402317  37252.26649295  null</p>
  <p>42575.1386371  37043.59910954  null</p>
  <p>42826.16706828  37026.34090489  null</p>
  <p>42766.49603263  37412.54781911  61.53707464</p>
</data_3d>
```

### 28.3.14 radius\_data and major\_data

If there are only straight and arc segments for the string, then for either **data\_2d** or **data\_3d**, it is possible to add a radius and major/minor arc flag for each segment of the string using the **radius\_data** and **major\_data** blocks respectively.

The order of the entries in the **radius\_data** and **major\_data** blocks must match the order of the segments in the string (which is also the order in the **data\_2d** or **data\_3d** block).

So there is exactly one entry for each segment.

**Note:** If there are n vertices in the super string, then there are (n-1) segments for a open string (not closed) and n segments for a closed string.

For each segment there are five possibilities for an arc going between the vertices and these are specified by using **positive**, **zero** or **negative** values for the **radius**, and **1** or **0** for the **major flag**.

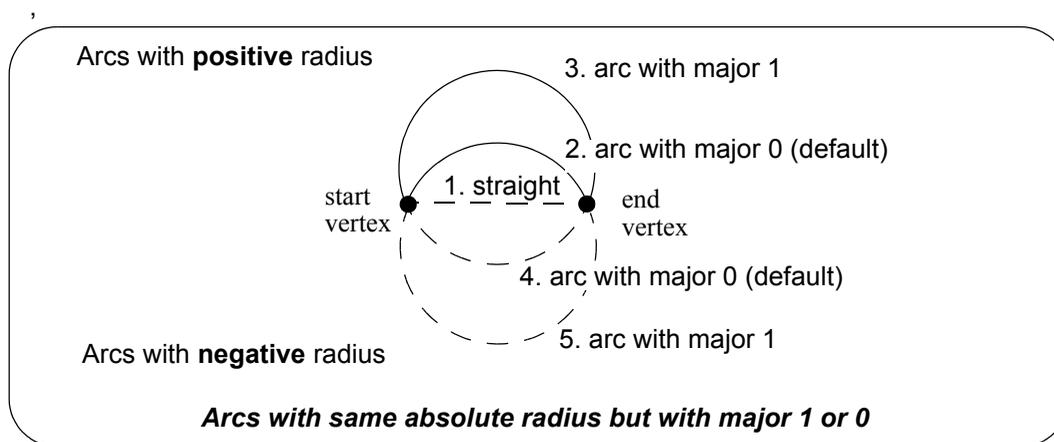
1. Straight segment - radius = 0. Major flag can be 1 or 0.
2. Positive radius and major flag 0
 

The arc is above the straight line joining the two vertices but the arc is the smaller of the two possibilities (minor arc).
3. Positive radius and major flag1
 

The arc is above the straight line joining the two vertices but the arc is the larger of the two possibilities (major arc).
4. Negative radius and major flag 0
 

The arc is below the straight line joining the two vertices but the arc is the smaller of the two possibilities (minor arc).
5. Negative radius and major flag1
 

The arc is below the straight line joining the two vertices but the arc is the larger of the two possibilities (major arc).



The **radius\_data** block is

```
<radius_data>
  radius_for_segment_1
  radius_for_segment_2
  ...
  radius_for_segment_m
</radius_data>
```

where

**radius\_for\_segment\_i** is the radius for the i'th segment and can be positive, zero or negative, and

$m = n-1$  for an open string or  $m = n$  for a closed string.

If the **radius\_block** is missing then the radius is taken to be 0 and all the segments are straight lines.

The **major\_data** block is

```
<major_data>
  major_flag_for_segment_1
  major_flag_for_segment_2
  ...
  major_flag_for_segment_m
</major_data>
```

where

**major\_flag\_for\_segment\_i** for the i'th segment is 1 or t if the arc is a major arc, and 0 or f if it is a minor arc, and

$m = n-1$  for an open string or  $m = n$  for a closed string.

If the **major\_block** is missing then the major flag is taken to be 0 and any segments with arcs are always the minor arcs.

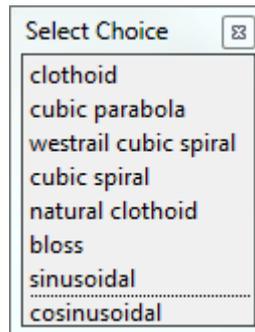
For example, for a closed string of five vertices

```
<radius_data>
  100 -300 0 0 0
</radius_data>
<major_data>
  f f f f f
```

```
</major_data>
```

## 28.3.15 Available Transition Types

The transition that are available are



where

**clothoid**, (or **spiral**) is the spiral approximation used by Australian road authorities and Queensland Rail

**cubic parabola** (or **state wide rail nsw**) is a special transition curve used by NSW Railways. It is not a spiral.

**westrail cubic spiral** (or **westrail-cubic**) is a spiral approximation used by WA railways.

**cubic spiral** (or **spiral**) is a low level spiral approximation. Mainly only used in surveying textbooks.

**natural clothoid** (or **landxml spiral** or **clothoid landxml**) is the full Euler spiral. Not used by any Authority in Australia or New Zealand.

**bloss** is a Bloss curve.

**sinusoidal** is a sinusoidal curve.

**cosinusoidal** is a cosinusoidal curve.

## 28.4 Attributes

Many **12d Model** objects (models and elements such as individual strings and tins) can have an unlimited number of named **attributes** of type integer (numbers), real and text. Super strings and drainage strings can also have attributes on each vertex and segment.

The attributes for an object are given in an **attributes block** which consists of the keyword **attributes** followed by the definitions of the *individual attributes* enclosed in start and end curly braces { and }. That is, an **attributes\_block** is

```
<attributes>
  attribute_1
  attribute_2
  ...
  attribute_n
</attributes>
```

where the attribute definitions for the individual attributes *attribute\_i* consists of

```
<attribute_type>
  <name> attribute_name </name>   <value> attribute_value </value>
</attribute_type>
```

where

*attribute\_type* is **integer**, **real** or **text**

*attribute\_name* is the unique attribute name for the object.

and

*attribute\_value* is the appropriate value of the integer, real or a text.

**OR**

where *attribute\_type* is **group**

```
<group>
  <name> group_name </name>   attributes_block
</group>
```

where

*group\_name* is the unique name of the group at this level

and

*attributes\_block* is another *attributes\_block*.

Note that the definition of **<group>** includes an *attribute\_block* which can contain another **<group>** so the definition is recursive.

Hence you can have a hierarchy or tree of attributes going down to any level.

Within an object, the attribute names are case sensitive and must be unique. That is, for attribute names, upper and lower case alphabet characters are considered to be different characters.

An example of an **attribute block** defining four attributes named "pole id", "street", "pole height" and "pole wires" is:

```
<attributes>
  <text> <name>pole id</name> <value>QMR-37</value> </text>
  <text> <name>street</name> <value>477 Boundary St</value> </text>
```

```
<real> <name>pole wires</name> <value>3</value> </text>  
</attributes>
```

Continue to the next section [28.5 Model](#) or return to [28 12d XML File Format](#).

## 28.5 Model

Within a **12d Model** project, information is collected in units called **MODELS**. The items that can be stored in a model are called elements and elements include strings, tins, super tins, grid tins, trimeshes and plot frames.

Each model has a unique user-defined text name, **model\_name**, of up to two hundred alphanumeric characters and spaces.

The format for the **model** keyword block is:

```
<model>
  <name>model_name</name>
  attribute_block
  time_created_block
  time_updated_block
  <children>
    element_data_1
    ...
    element_data_n
  </children>
</model>
```

where:

*model\_name* is a string of characters for the model name. For the characters allowed, see

*attribute\_block* is option. For *attributes\_block* see [28.4 Attributes](#).

*time\_created\_block* is optional. See [28.3.7 Time Created](#).

*time\_updated\_block* is optional. See [28.3.8 Time Updated](#).

*element\_data\_i* is an element stored in the model. See [28.6 Elements Contained in Models](#).

The **children** block is **optional** and is mainly there so that in an xml editor, the *element\_data\_i* items can be collapsed into the **children** section.

An example of a model with no elements and no *children* block:

```
<model>
  <name>telegraph poles,/name>
  <attributes>
    <text> <name>pole id</name> <value>QMR-37</value> </text>
    <text> <name>street</name><value>477 Boundary St</value></text>
    <real> <name>pole wires</name> <value>3</value> </text>
  </attributes>
</model>
```

Continue to the next section [28.6 Elements Contained in Models](#) or return to [28 12d XML File Format](#).

## 28.6 Elements Contained in Models

See

[28.6.1 Tin](#)

[28.6.2 Super Tin](#)

[28.6.5 Arc String](#)

[28.6.6 Circle String](#)

[28.6.7 Drainage String](#)

[28.6.8 Feature String](#)

[28.6.9 Plot Frame String](#)

[28.6.10 Super String](#)

[28.6.11 Super Alignment String](#)

[28.6.12 Text String](#)

## 28.6.1 Tin

A *tin* (triangulated *irregular networks*) is an element that may, or may not, be in a model.

Each tin has text name, ***tin\_name***, of up to two hundred alphanumeric characters and spaces and the names of each tin and super tin in the project must be unique.

The format for the **tin** element is:

```
<tin>
  <name>tin_name</name>
  attribute_block
  time_created_block
  time_updated_block
  colour_block
  points_block
  triangles_block
  colours_block
  input_block
</tin>
```

where

### ***tin\_name***

is a string of characters for the tin name and can't be blank. This must be unique in a project.

For the characters that can make up a *tin\_name*, see [Names of models, tins, styles, colours and attributes](#).

### ***time\_created\_block***

is the time the tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

### ***time\_updated\_block***

is the last time the tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

### ***colour\_block***

this colour number is the primary (base) colour for all the triangles in the tin. A triangle in the tin will have this colour unless it is overridden by a *colours\_block*. For the syntax of *colour\_block*, see [28.3.2 Colour](#).

***attribute\_block*** is optional: For the syntax of an *attributes\_block* see [28.4 Attributes](#).

The attributes in this block and the *attributes\_block* itself are optional.

The attributes **Style**, **Weed**, **Faces**, **Boundary\_String**, **null\_length**, **null\_angle**, **null\_combined\_length** and **null\_combined\_angle** are special attributes that have extra information used by **12d Model** to create the tin. These special attributes should not be deleted.

The format of the special attributes inside the **<attributes> ... </attributes>** is:

```
<text> <name>Style</name> <value>style_name</value> </text>
<integer> <name>Weed</name> <value>weed_value</value> </integer>
<integer> <name>Faces</name> <value>faces_value</value> </integer>
<text> <name>Boundary_String</name><value>full_string_name</value></text>
<real> <name>null_length</name> <value>>null_len_val</value> </real>
<real> <name>null_angle</name> <value>>null_angle_rad</value> </real>
<real> <name>null_combined_length</name> <value>>null_com_ln/value> <real>
```

```
<real> <name>null_combined_angle</name><value>null_com_rad</value></real>
```

where

**style\_name** is the style for the tin

**weed\_value** is 0 or 1

**faces\_value** is 0 if the data is not from triangles, 1 if the data is from triangles

**full\_string\_name** is the name of a polygon for nulling outside. This is optional.

**null\_len\_val** is value for nulling by angle

**null\_angle\_rad** is in radians value for nulling by angle

**null\_com\_In** is for nulling by combined angle and length

**null\_com\_rad** is in radians for nulling by combined angle and length

### **points\_block**

This gives the coordinates of the points that will be vertices of the triangles in the tin. The points are implicitly numbered by the order in the list (starting at point 1). The Points Block is MANDATORY.

```
<points>
```

```
<p>x_value_1 y_value_1 z_value_1</p>
```

```
<p>x_value_2 y_value_2 z_value_2</p>
```

```
...
```

```
<p>x_value_m y_value_m z_value_m</p>
```

```
</points>
```

where (x\_value\_j, y\_value\_j, z\_value\_j) are the coordinates of the j'th point.

### **triangles\_block**

This gives the triangles that make up the tin.

Each triangle is given as a triplet of the point numbers in the Points block that are the triangle vertices. The order of the triangles is unimportant. The Triangles Block is MANDATORY

```
<triangles>
```

```
<t>t1_pt_1 t1_pt_2 t1_pt_3</t>
```

```
<t>t2_pt_1 t2_pt_2 t2_pt_3</t>
```

```
...
```

```
<t>tn_pt_1 tn_pt_2 tn_pt_3</t>
```

```
</triangles>
```

where *tk\_pt\_1* *tk\_pt\_2* *tk\_pt\_3* are point numbers from the points\_block of the three vertices of the k'th triangle.

### **colours\_block**

Triangles can be given colours other than the base colour by including a Colours Block. The colour for each triangle is then individually given where -1 means use the base colour. The order of the entries in the colours block must match the order of the triangles in the Triangles Block. So there is exactly one entry for each triangle

If all the triangles are the base colour, then the *Colours Block* is omitted.

```
<colours>
```

```

c1  c2  ... c15 c16
c17 c18  ... c31 c32
...
cn-2 cn-1  cn

```

**</colours>**

where **ck** is the colour number of the k'th triangle in the *triangles\_block*.

**ck** equals -1 when there is no special colour set and the triangle is drawn in the base colour.

### **input\_block**

The *input\_block* gives more information about how the tin was created by 12d Model.

None of this information is needed when reading a tin into 12d Model and the *input\_block* can be omitted.

**<input>**

**<preserve\_strings>** *pres\_str\_text\_logical* **</preserve\_strings>**

**<remove\_bubbles>** *rem\_bub\_text\_logical* **</remove\_bubbles>**

**<weed\_tin>** *weed\_tin\_text\_logical* **</weed\_tin>**

**<triangle\_data>** *triangle\_data\_text\_logical* **</triangle\_data>**

**<sort\_tin>** *sort\_tin\_text\_logical* **</sort\_tin>**

**<cell\_method>** *cell\_method\_text\_logical* **</cell\_method>**

**<models>**

*model\_name\_1*

*model\_name\_2*

...

*model\_name\_p*

**</models>**

**</input>**

where

*pres\_str\_text\_logical*, *rem\_bub\_text\_logical*, *weed\_tin\_text\_logical*, *triangle\_data\_text\_logical*, *sort\_tin\_text\_logical* and *cell\_method\_text\_logical* are text and can only have the values **true** or **false**.

**<models> ... </models>** is the list of models in the tin where

*model\_name\_i* is the name of the i'th model making up the tin.

Continue to the next section [28.6.2 Super Tin](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

## 28.6.2 Super Tin

A *Super Tins* consists of a number of **tins** (triangulated *irregular networks*).

Each super tin has text name, **tin\_name**, of up to two hundred alphanumeric characters and spaces and the names of each tin and super tin in the project must be unique.

The format for the **super\_tin** element is:

```
<super_tin>
  <name>tin_name</name>
  attribute_block
  time_created_block
  time_updated_block
  colour_block
  exact_block
  tins_block
</super_tin>
```

where

### **tin\_name**

is a string of characters for the super tin name and can't be blank. This must be unique in a project.

For the characters that can make up a tin\_name, see [Names of models, tins, styles, colours and attributes](#).

### **time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

### **time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

### **colour\_block**

this colour number is the primary (base) colour for the super tin. For the syntax of colour\_block, see [28.3.2 Colour](#).

**attribute\_block** is optional: For the syntax of an *attributes\_block* see [28.4 Attributes](#).

The attributes in this block and the *attributes\_block* itself are optional.

The attribute **Style** is a special attribute that is used by **12d Model** to create the super tin. This special attribute should not be deleted.

The format of the **Style** attribute inside the <attributes> ... </attributes> is:

```
<text> <name>Style</name> <value>style_name</value> </text>
```

where

**style\_name** is the style for the super tin

### **exact\_block**

```
<exact> exact_text_logical </exact>
```

where

**exact\_text\_logical** is text and can only have the value **true** or **false**.

**tins\_block**

This gives the tins that make up the super tin within the keyword block **tins**.

```
<tins>
  tin_info_1
  tin_info_2
  ...
  tin_info_p
</tins>
```

where

there are *p* tins in the super tin and *tin\_info\_i* is information about the *i*'th tin. The information about a tin is contained in a **tin** block.

```
<tin>
  <name> tin_name_i</name>
  <active> active_text_logical</active>
  <mode> mode_text_logical</mode>
</tin>
```

where

**tin\_name\_i** is the name of the *i*'th tin making up the super tin.

*active\_text\_logical* and *mode\_text\_logical* are text and can only have the value **true** or **false**.

For example

```
<super_tin>
  <name>super tin</name>
  <colour>green</colour>
  <attributes>
    <text> <name>Style</name> <value>1</value> </text>
  </attributes>
  <time_created>28-Apr-2015 06:42:45</time_created>
  <time_updated>28-Apr-2015 06:42:45</time_updated>
  <exact>true</exact>
  <tins>
    <tin>
      <name>DESIGN ALL</name>
      <active>true</active>
      <mode>replace</mode>
    </tin>
    <tin>
      <name>HILL</name>
      <active>true</active>
      <mode>replace</mode>
    </tin>
  </tins>
</super_tin>
```

Note that the tins that make up the super tin must exist in **12d Model** for the super tin to be fully defined.

Continue to the next section [28.6.3 String Header Block](#) or return to [28.3 Regularly Used Keyword Blocks](#) or [28 12d XML File Format](#).

## 28.6.3 String Header Block

Strings are special types of elements that reside in a model.

Strings have common header information and this will be documented in this one spot as a ***string\_header\_block***.

The format for the `string_header_block` is:

```

string_name_block
chainage_block
colour_block
style_block
weight_block
interval_block
time_created_block
time_updated_block
attribute_block

```

where

### ***string\_name\_block***

The format of the ***string\_name\_block*** is:

```
<name> string_name_text </name>
```

where

***string\_name\_text*** is a string of allowable characters that is the name of the string.

For the characters that can make up a `string_name`, see [String names](#).

Any leading and trailing spaces will be removed in the string name.

`string_name` can be blank.

An example of a string name is:

```
<name> design 100.0 </name>
```

### ***chainage\_block***

is the start chainage of the string. This is optional. For the syntax see [28.3.4 Chainage](#).

### ***colour\_block***

the colour name is the primary colour for the string. For the syntax of `colour_block`, see [28.3.2 Colour](#).

### ***style\_block***

is the line style of the string. This is optional. For the syntax of `style_block` see [28.3.3 Line Style](#).

### ***weight\_block***

is the weight (thickness) of the string. This is optional. For the syntax of `weight_block` see [28.3.5 Weight](#).

### ***interval\_block***

the chainage interval to temporarily introduce extra vertices into the string when the string is in a triangulation to form a tin. For the syntax of `interval_block`, see [28.3.6 Interval](#).

### ***time\_created\_block***

is the time the super tin was originally created, This is optional. For the syntax of `time_created_block` see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax of time\_updated\_block see [28.3.8 Time Updated](#).

**attribute\_block**

The string attributes are in this block. For the syntax of an *attributes\_block* see [28.4 Attributes](#)

The attributes\_block is optional.

For example

**string\_header\_block** →

```
<string_arc>
  <name>arc</name>
  <chainage>0</chainage>
  <breakline>line</breakline>
  <colour>yellow</colour>
  <style>1</style>
  <weight>2</weight>
  <time_created>28-Apr-2015 07:46:57</time_created>
  <time_updated>28-Apr-2015 07:46:57</time_updated>
  <interval>10</interval>
  <centre>1067.40263766 530.14953857 0</centre>
  <radius>226.6814323</radius>
  <chord_arc>0.1</chord_arc>
  <start>867.42825529 423.40349345 0</start>
  <end>1118.02452861 751.10631241 0</end>
</string_arc>
```

Continue to the next section [28.6.4 Text Information](#) or return to [28.6 Elements Contained in Models](#) or [28 12d XML File Format](#).

## 28.6.4 Text Information

See

[28.6.4.1 Vertex Annotation Information](#)

[28.6.4.2 Segment Annotation Information](#)

### 28.6.4.1 Vertex Annotation Information

The *vertex\_annotation\_information* is

```
<worldsize> world_size_real </worldsize>
<textstyle> textstyle_name </textstyle>
<angle> angle_dec_deg_real </angle>
<x_factor> x_factor_real </x_factor>
<slant> slant_dec_deg_real </slant>
<offset> offset_real </offset>
<raise> raise_real </raise>
<text_colour> text_colour_name </text_colour>
<justify> text_justification_text </justify>
```

where

***world\_size\_real*** is the size of the text in world units.

***textstyle\_name*** is the name of the textstyle for the text.

***angle\_dec\_deg\_real*** is the angle of the text. The value is in decimal degrees and is measured in a counter clockwise direction from the positive x-axis.

***x\_factor\_real*** is the factor to apply to the width of the text.

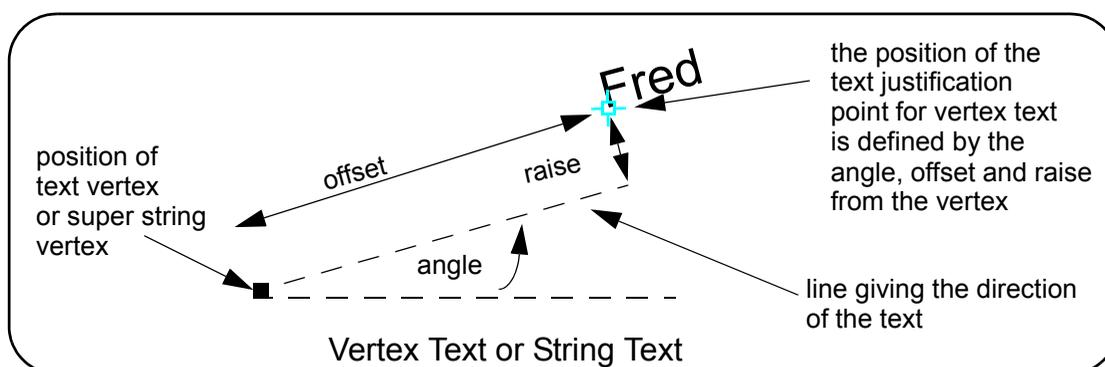
***slant\_dec\_deg\_real*** is the angle the text is slanted from the vertical. The value is in decimal degrees and is measured in a clockwise direction from the positive y-axis.

***offset\_real*** is distance to offset the text from the text vertex.

***raise\_real*** is the perpendicular distance the text is off the direction line of the text.

***text\_colour\_name*** is the colour of the text. This should be the same as the colour in the *string\_header\_block*. For the syntax of *colour\_block*, see [28.3.2 Colour](#).

***text\_justification\_text*** is the text giving the justification point of the text.



### 28.6.4.2 Segment Annotation Information

The *segment\_annotation\_information* is

```

<worldsize> world_size_real </worldsize>
<textstyle> textstyle_name </textstyle>
<angle> angle_dec_deg_real </angle>
<x_factor> x_factor_real </x_factor>
<slant> slant_dec_deg_real </slant>
<offset> offset_real </offset>
<raise> raise_real </raise>
<text_colour> text_colour_name </text_colour>
<justify> text_justification_text </justify>

```

where

*world\_size\_real* is the size of the text in world units.

*textstyle\_name* is the name of the textstyle for the text.

*angle\_dec\_deg\_real* is the angle of the text. The value is in decimal degrees and is measured in a counter clockwise direction from the segment.

*x\_factor\_real* is the factor to apply to the width of the text.

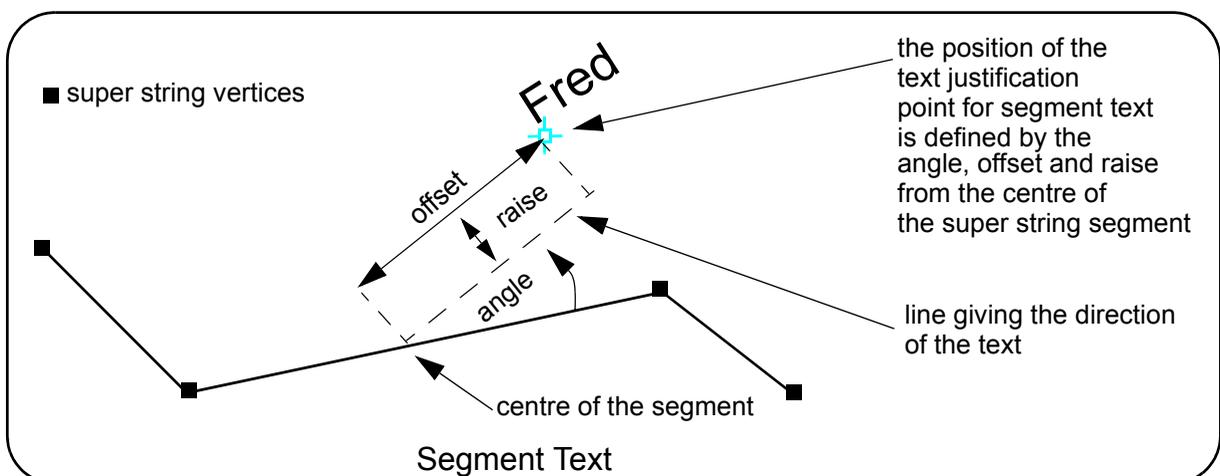
*slant\_dec\_deg\_real* is the angle the text is slanted from the vertical. The value is in decimal degrees and is measured in a clockwise direction from the positive y-axis.

*offset\_real* is distance to offset the text from the centre of the segment.

*raise\_real* is the perpendicular distance the text is off the direction line of the text.

*text\_colour\_name* is the colour of the text. This should be the same as the colour in the *string\_header\_block*. For the syntax of *colour\_block*, see [28.3.2 Colour](#).

*text\_justification\_text* is the text giving the justification point of the text.



## 28.6.5 Arc String

The format for the **string\_arc** element is:

```
<string_arc>
  string_header_block
  centre_block
  radius_block
  chord_arc_block
  start_block
  end_block
</string_arc>
```

where

### **string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

### **centre\_block**

The format of the **centre\_block** is:

```
<centre> x_centre_real y_centre_real z_centre_real </centre>
```

where

$(x\_centre\_real, y\_centre\_real, z\_centre\_real)$  is the centre of the arc.

### **radius\_block**

the radius of the arc. For the syntax of **radius\_block**, see [28.3.9 Breakline](#).

A positive radius means that the arc goes from the start point in a clockwise direction (goes to the right) and a negative radius means that the arc goes is in a counter clockwise direction (goes to the left).

### **chord\_arc\_block**

The format of the **chord\_arc\_block** is:

```
<chord_arc>chord_arc_real </chord_arc>
```

where

**chord\_arc\_real** is a real number and is the chord to arc tolerance to use to temporarily insert vertices into the arc when the arc is included in a triangulation to form a tin.

### **start\_block**

The format of the **start\_block** is:

```
<start> x_start_real y_start_real z_start_real </start>
```

where

$(x\_start\_real, y\_start\_real, z\_start\_real)$  is the start coordinate of the arc.

### **end\_block**

The format of the **end\_block** is:

```
<end> x_end_real y_end_real z_end_real </end>
```

where

$(x\_end\_real, y\_end\_real, z\_end\_real)$  is the end coordinate of the arc.

For example

```
<string_arc>
  <name>arc</name>
  <chainage>0</chainage>
  <breakline>line</breakline>
  <colour>yellow</colour>
  <style>1</style>
  <weight>2</weight>
  <time_created>28-Apr-2015 07:46:57</time_created>
  <time_updated>28-Apr-2015 07:46:57</time_updated>
  <interval>10</interval>
  <centre>1067.40263766 530.14953857 0</centre>
  <radius>226.6814323</radius>
  <chord_arc>0.1</chord_arc>
  <start>867.42825529 423.40349345 0</start>
  <end>1118.02452861 751.10631241 0</end>
</string_arc>
```

Continue to the next section [28.6.6 Circle String](#) or return to [28.6.3 String Header Block](#) or [28 12d XML File Format](#).

## 28.6.6 Circle String

The format for the **string\_circle** element is:

```
<string_circle>
  string_header_block
  centre_block
  radius_block
  chord_arc_block
</string_arc>
```

where

### **string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

### **centre\_block**

The format of the **centre\_block** is:

```
<centre> x_centre_real y_centre_real z_centre_real </centre>
```

where

(*x\_centre\_real*,*y\_centre\_real*,*z\_centre\_real*) is the centre of the circle.

### **radius\_block**

the radius of the circle. For the syntax of *radius\_block*, see [28.3.9 Breakline](#).

A positive radius means that the circle goes in a clockwise direction (goes to the right) and a negative radius means that the circle goes is in a counter clockwise direction (goes to the left).

### **chord\_arc\_block**

The format of the **chord\_arc\_block** is:

```
<chord_arc>chord_arc_real </chord_arc>
```

where

**chord\_arc\_real** is a real number and is the chord to arc tolerance to use to temporarily insert vertices into the circle when the circle is included in a triangulation to form a tin.

For example

```
<string_circle>
  <name>circle</name>
  <chainage>0</chainage>
  <breakline>line</breakline>
  <colour>yellow</colour>
  <style>1</style>
  <weight>5</weight>
  <interval>10</interval>
  <time_created>28-Apr-2015 07:45:53</time_created>
  <time_updated>28-Apr-2015 07:46:23</time_updated>
  <centre>409.93551 548.76354 null</centre>
  <radius>100</radius>
  <chord_arc>0.1</chord_arc>
</string_circle>string circle
```

Continue to the next section [28.6.7 Drainage String](#) or return to [28.6.3 String Header Block](#) or [28.12d XML File Format](#).

## 28.6.7 Drainage String

The full 12dXML definition of the drainage string is:

```
<string_drainage>
  string_header_block
  outfall_block
  flow_direction_block
  use_pit_con_points_block
  drainage_sewer_block
  data_3d_block
  radius_data_block
  major_data_block
  pit_records
  pipe_records
</string_drainage>
```

where

### **string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

There are also some special attributes in the string attributes in the String Header Block that provide extra information for the drainage string.

### **outfall\_block**

```
<outfall> outfall_real </outfall>
```

where *outfall\_real* is the z-value of the outfall (the low end of the string).

### **flow\_direction\_block**

```
<flow_direction> flow_direction_flag </flow_direction>
```

where *flow\_direction\_flag* is **1** if the flow is the same as the string direction, or **0** if the flow is opposite to the string direction.

### **use\_pit\_con\_points\_block**

```
<user_pit_con_points> use_pit_connection_points_logical_text </user_pit_con_points>
```

where *use\_pit\_connection\_points\_logical\_text* is **true** if pit connection points are used, or **false** if pit connection points are not being used and hence the pipes go to the centre of the pits.

### **drainage\_sewer\_block**

```
<drainage_sewer> drainage_sewer_choice_text </drainage_sewer>
```

where *drainage\_sewer\_choice\_text* is **drainage** (storm water) if it is for drainage and **sewer** if it is for sewer (foul water).

### **data\_3d\_block, radius\_data\_block and major\_data\_block**

the drainage string has an underlying string that is used to define locations of the pits and the geometry for the pipes. The underlying string can have straight and arc segments.

The vertex data for the underlying string is given in a **data\_3d** block, and if there any arcs, then these are specified in **radius\_data** and **major\_data** blocks. See [28.3.13 data\\_3d](#) and [28.3.14 radius\\_data and major\\_data](#).

**pit\_records**

In plan the pits sit on the underlying string and there is one pit record for each pit. The pits do not have to be on a vertex of the underlying string.

There is one **pit** block for each pit in the string and they are in the order that they occur along the string.

The information for each **pit** is:

```

<pit>
  <name> pit_name_text </name>
  <type> pit_type_text </type>
  <chainage> pit_chainage_real </chainage>
  <ip> pit_ip_text </ip>
  <ratio> pit_ratio_real </ratio>
  <x> pit_x_real </x>
  <y> pit_y_real </y>
  <z> pit_z_real </z>
  <road_chainage> pit_road_chainage_real </road_chainage>
  <diameter> pit_diameter_real </diameter>
  <width> pit_width_real </width>
  <sump_level> pit_sump_level_real </sump_level>
  <floating_sump> pit_floating_sump_flag </floating_sump>
  <thickness> pit_thickness_real </thickness>
  <thickness_bottom> pit_thickness_bottom_real </thickness_bottom>
  <thickness_back> pit_thickness_back_real </thickness_back>
  <thickness_left> pit_thickness_left_real </thickness_left>
  <thickness_right> pit_thickness_right_real </thickness_right>
  <con_point_mode> pit_con_points_mode_text </con_point_mode>
  <floating> pit_floating_logical_text </floating>
  <hgl> pit_hgl_real </hgl>
  pit_attributes_block
</pit>

```

where

**pipe\_records**

In plan the pipes sit on the underlying string and the plan geometry is based on the underlying string. Each pipe goes between two adjacent pits.

There is one **pipe** block for each pipe in the string and they are in the order that they occur along the string.

```

<pipe>
  <name> pipe_name_text </name>
  <type> pipe_type_text </type>
  <colour> pipe_colour_text </colour>
  <diameter> pipe_diameter_real </diameter>

```

```

<nominal_diameter> pipe_nominal_diameter_real </nominal_diameter>
<width> pipe_width_real </width>
<top_width> pipe_top_width_real </top_width>
<thickness> pipe_thickness_real </thickness>
<thickness_bottom> pipe_thickness_bottom_real </thickness_bottom>
<thickness_back> pipe_thickness_back_real </thickness_back>
<thickness_left> pipe_thickness_left_real </thickness_left>
<thickness_right> pipe_thickness_right_real </thickness_right>
<separation> pipe_separation_real </separation>
<number_of_pipes> pipe_number_of_pipes_integer </number_of_pipes>
<us_level> pipe_us_level_real </us_level>
<ds_level> pipe_ds_level_real </ds_level>
<us_hgl> pipe_us_hgl_real </us_hgl>
<ds_hgl> pipe_ds_hgl_real </ds_hgl>
<flow_velocity> pipe_flow_velocity_real </flow_velocity>
<flow_volume> pipe_flow_volume_real </flow_volume>
pipe_attributes_block
</pipe>

```

```

string drainage {
  chainage start_chainage
  model model_name name string_name
  colour colour_name style style_name
  breakline point or line
  attributes {
    text Tin finished_surface_tin
    text NSTin natural_surface_tin
    integer "_floating" 1|0 // 1 for floating, 0 not floating
  }
  outfall outfall_value // z-value at the outfall
  flow_direction 0|1 // 0 drainage line is defined from downstream
  // to upstream

  data { // key word - geometry of the drainage string
    x-value y-value z-value radius bulge
    " " " "
    " " " "
  }
  pit { // pit/manhole - one pit record for each pit/manhole
    // in the order along the string
    name text // pit name
    type text // pit type
    road_name text // road name
    road_chainage chainage // road chainage
    diameter value // pit diameter
    floating yes|no // is pit floating or not
    chainage pit_chainage // internal use only
  }
}

```

```

    ip            value           // internal use only
    ratio        value           // internal use only
    x            x-value         // x-value of top of pit
    y            y-value         // y-value of top of pit
    z            z-value         // z-value of top of pit
}
pipe { // one pipe record for each pipe connecting pits/manholes
// in the order they occur along the string
    name        text           // pipe name
    type        text           // pipe type
    diameter    value          // pit diameter
    us_level    value          //
    ds_level    value          //
    us_hgl      value          //
    ds_hgl      value          //
    flow_velocity value       //
    flow_volume value         //
}
property_control {
    name        text           // lot name
    colour      colour_name
    grade       value          // grade of pipe in units of "1v in"
    cover       value          // cover of the of pipe
    diameter    value          // diameter of the of pipe
    boundary    value          // boundary trap value
    chainage    chainage       // internal use only
    ip          value          // internal use only
    ratio       value          // internal use only
    x           x-value        // x value of where pipe connects to sewer
    y           y-value        // y value of where pipe connects to sewer
    z           z-value        // internal use only

    data { // key word - geometry of the property control
        x-value  y-value  z-value  radius  bulge
        "        "        "
        "        "        "
    }
}
house_connection { //warning - house connections may change in future versions
    name        text           // house connection name
    hcb         integer        // user given integer
    colour      colour_name
    grade       value          // grade of connection in units of "1v in"
    depth       value
    diameter    value
    side        left or right
    length      value
    type        text           // connection type
    material    text           // material type
    bush        text           // bush type
    level       value
    adopted_level value
    chainage    chainage       // internal use only
    ip          value          // internal use only
    ratio       value          // internal use only
    x           x-value        // x value of where pipe connects to sewer
    y           y-value        // y value of where pipe connects to sewer
    z           z-value        // internal use only
}

```

```
}           // end of drainage-sewer data
```

Continue to the next section [28.6.8 Feature String](#) or return to [28.6.3 String Header Block](#) or [28.12d XML File Format](#).

## 28.6.8 Feature String

The full 12dXML definition of the drainage string is:

```
<string_feature>
  string_header_block
  <radius> feature_radius_real </radius>
  <centre> x_centre_real y_centre_real z_centre_real </centre>
</string_feature>
```

where

### **string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

There are also some special attributes in the string attributes in the String Header Block that provide extra information for the drainage string.

**feature\_radius\_real** is the radius of the feature string.

**(xcentre\_real, y\_centre\_real, z\_centre\_real)** is the centre of the feature string.

For example

```
<string_feature>
  <name>Line 1</name>
  <chainage>0</chainage>
  <breakline>line</breakline>
  <colour>cyan</colour>
  <style>1</style>
  <time_created>2015-05-19T08:06:01Z</time_created>
  <time_updated>2015-05-19T08:06:01Z</time_updated>
  <centre>42200.06055 37384.05873 null</centre>
  <radius>20</radius>
</string_feature>
```

Continue to the next section [28.6.9 Plot Frame String](#) or return to [28.6.3 String Header Block](#) or [28 12d XML File Format](#).

## 28.6.9 Plot Frame String

The format for the **string\_plot\_frame** element is:

```
<string_plot_frame>
  info_block
  time_created_block
  time_updated_block
  sheet_details_block
  title_block_block
  origin_block
  scale_block
  rotation_block
  plotter_details_block
</string_plot_frame>
```

where

### **info\_block**

The format of the **info\_block** is:

```
<info>
  <name> plot_frame_name_text</name>
  <colour> plot_frame_name_colour_text</colour>
  <plot_file> plot_file_name_text</plot_file>
</info>
```

where

*plot\_frame\_name\_text* is a string of allowable characters that is the name of the plot file string. For the characters that can make up a string\_name, see [String names](#).

*plot\_frame\_colour\_text* is the colour of the plot frame. For the syntax of colour\_block, see [28.3.2 Colour](#).

*plot\_file\_name\_text* is the name of file that the plot frame will plot to.

### **time\_created\_block**

is the time the plot frame was originally created, This is optional. For the syntax of the time\_created\_block see [28.3.7 Time Created](#).

### **time\_updated\_block**

is the last time the plot frame was last modified, This is optional. For the syntax of the time\_updated\_block see [28.3.8 Time Updated](#).

### **sheet\_details\_block**

The format of the **sheet\_details\_block** is:

```
<sheet_details>
  <sheet_code> sheet_code_text</sheet_code>
  <width> sheet_width_real</width>
  <height> sheet_height_real</height>
  <left_margin> sheet_left_margin_real</left_margin>
  <right_margin> sheet_right_margin_real</right_margin>
  <top_margin> sheet_top_margin_real</top_margin>
```

```

<bottom_margin> sheet_bottom_margin_real</bottom_margin>
<border> sheet_border_text_logical</border>
<viewport> sheet_viewport_text_logical</viewport>

```

### </sheet\_details>

where

*sheet\_code\_text* is the name of the sheet. This can be blank.

*sheet\_width\_real*, *sheet\_height\_real*, *sheet\_left\_margin\_real*, *sheet\_right\_margin\_real*, *sheet\_top\_margin\_real*, *sheet\_bottom\_margin\_real* are all real values and give the size and margins for the sheet that the plot frame will plot. The units for all of them is millimetres.

*plot\_frame\_border\_text\_logical* and *plot\_frame\_viewport\_text\_logical* are text and can only have the value **true** or **false**.

### **origin\_block**

The format of the **origin\_block** is:

```
<origin> x_real y_real z_real</origin>
```

where

(*x\_real*,*y\_real*,*z\_real*) is the coordinates of the origin of the plot frame.

### **scale\_block**

The format of the **scale\_block** is:

```
<scale> scale_real</scale>
```

where

*scale\_real* is the 1: scale for the plots created by the plot frame.

### **rotation\_block**

The format of the **rotation\_block** is:

```
<rotation> rotation_dec_deg_real</rotation>
```

where

*rotation\_dec\_deg\_real* is rotation of the plot frame. The value is in decimal degrees and is measured in a counter clockwise direction from the positive x-axis.

### **plotter\_details\_block**

The format of the **plotter\_details\_block** is:

#### <plotter\_details>

```
<title_1>title_1_text</title_1>
```

```
<title_2>title_2_text</title_2>
```

```
<use_title_file> title_file_text_logical</border>
```

```
<title_file> title_file_name_text</title_file>
```

```
<text_size> title_text_size_real_mm</text_size>
```

```
<textstyle> title_text_style</textstyle>
```

#### </plotter\_details>

where

*title\_1\_text* and *title\_2\_text* are two lines of text for the title block. They can be blank.

*use\_title\_file\_text\_logical* is text and can only have the value **true** or **false**.

*title\_file* is the path name of the file to use as a title block file. This can be blank.

*title\_text\_size\_real\_mm* is the size of the text in the title block. The units are millimetres.

For example

```
<string_plot_frame>
  <info>
    <name>Plot frame</name>
    <colour>green</colour>
    <plot_file>plot</plot_file>
  </info>
  <sheet_details>
    <sheet_code>A0</sheet_code>
    <width>1189</width>
    <height>841</height>
    <left_margin>5</left_margin>
    <right_margin>10</right_margin>
    <bottom_margin>5</bottom_margin>
    <top_margin>10</top_margin>
    <border>true</border>
    <viewport>true</viewport>
  </sheet_details>
  <title_block>
    <title_1>Title 1</title_1>
    <title_2>Title 2</title_2>
    <use_title_file>true</use_title_file>
    <title_file>A0 title.tbf</title_file>
    <text_size>5</text_size>
    <textstyle>1</textstyle>
  </title_block>
  <origin>695.2353 1464.6248</origin>
  <scale>100</scale>
  <rotation>45</rotation>
  <plotter_details>
    <id>9</id>
    <type>model</type>
    <mode>""</mode>
    <names>""</names>
  </plotter_details>
  <time_created>29-Apr-2015 01:11:52</time_created>
  <time_updated>29-Apr-2015 01:11:52</time_updated>
</string_plot_frame>
```

Continue to the next section [28.6.10 Super String](#) or return to [28.6.3 String Header Block](#) or [28.12d XML File Format](#).

## 28.6.10 Super String

Because the super string is so versatile, its 12d XML format looks complicated but it is very logical and actually quite simple.

In its most primitive form, the super string is simply a set of (x,y) values as in a 2d string, or (x,y,z) values as in a 3d string.

Additional blocks of information can extend the definition of the super string and only need to be included if they exist. For example, segment arcs or transitions, vertex ids, vertex and segment text, round pipe diameters or box pipes widths and heights and tinability.

Some of the properties of the super string can be constant for the entire string or can vary for each vertex and/or segment. For example, there can be one colour for the entire string or individual colours for each segment.

For user attributes, the super string not only has the standard user attributes defined for the entire string (string attributes), but also can have user attributes for each vertex (vertex attributes) and for each segment (segment attributes).

Being closed or not is another property of the super string and if the super string is closed then the super string knows there is an additional segment going from the last vertex back to the first vertex. This means that there is no duplication of the first and last vertex needed.

Thus if a super string has  $n$  vertices, then an open super string has  $n-1$  segments joining the vertices and a closed super string has  $n$  segments since there is an additional segment from the last to the first vertex.

With the additional data for vertices and/or segments in the super string, the data is in vertex or segment order.

So for a string with  $n$  vertices, there must be  $n$  bits of vertex data.

For segments, if the string is open then there only needs to be  $n-1$  bits of segment data but for closed strings, there must be  $n$  bits of data.

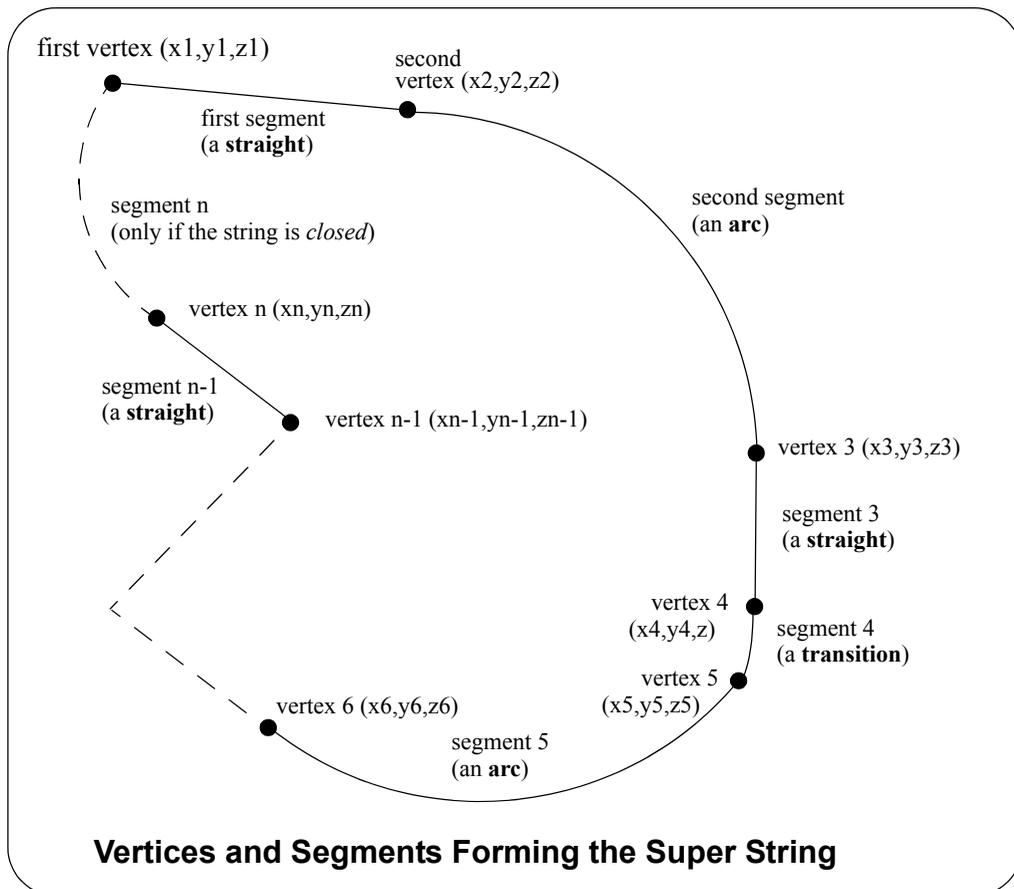
For an open string,  $n$  bits of segment data can be specified and the  $n$ th bit will be read in and stored. If the string is then closed, the  $n$ th bit of data will be used for the extra segment.

### Important Note

For a super string, the arcs, transitions and offset transitions are that shape in plan.

Hence an arc with a z-value at each end is actually a **helix** and **NOT** part of a three dimensional circle.

Transitions and offset transitions are helixes with a constantly changing radius.



The 12dXML definition of the super string is:

```

<string_super>
  string_header_block
  closed_block
  interval_block
  blocks_of_info_1
  blocks_of_info_2
  ...
  blocks_of_info_n
</string_super>

```

where

**string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

**closed\_block**

```
<closed> closed_text_logical </closed>
```

where *closed\_text\_logical* is **true** if the super string is closed and **false** if the super string is open.

**interval\_block**

The `interval_block` for a super string has a *distance* (a chainage interval) and a *chord\_to\_arc\_real*

where

the ***distance*** to temporarily introduce extra vertices into the string at the given chainage distance when the string is in a triangulation to form a tin.

***chord\_arc\_real*** is a real number and is the chord to arc tolerance to use on any arcs in the super string to temporarily insert vertices into the arc when the arc is included in a triangulation to form a tin.

For the syntax of `interval_block`, see [28.3.6 Interval](#).

#### ***blocks\_of\_info***

The blocks of info can be broken up into four types.

- (a) blocks defining the position of the vertices in z, y and z

Each vertex must have at least an (x,y) value but there may be one z-value for the entire string and (x,y) at each vertex (`data_2d`), or an (x,y,z) for each vertex (`data_3d`).

See [28.6.10.1 Defining the Coordinates of the Vertices](#)

- (b) blocks defining the geometry of the segments

Segments can be straights, arcs, transitions or offset transitions.

*radius\_data* and *major\_data* or *geometry\_data*.

See [28.6.10.2 Geometry of the Horizontal Segments](#)

- (c) extra information for the vertices and/or segments such as colour, attributes, vertex ids, symbols tinability etc.

The definition for the blocks of each type now follows.

[28.6.10.1 Defining the Coordinates of the Vertices](#)

[28.6.10.2 Geometry of the Horizontal Segments](#)

[28.6.10.3 Colour](#)

[28.6.10.4 String, Vertex and Segment Attributes](#)

[28.6.10.5 Vertex Id's \(Point Id's\)](#)

[28.6.10.6 Symbols at Vertices](#)

[28.6.10.7 Tinability](#)

[28.6.10.8 Round or Box \(Culvert\) Pipes](#)

[28.6.10.9 Vertex Text and Vertex Annotation](#)

[28.6.10.10 Segment Text and Segment Annotation](#)

## 28.6.10.1 Defining the Coordinates of the Vertices

See

[28.6.10.1.1 One Z or No Z for the String](#)

[28.6.10.1.2 Varying Z Values along the String](#)

### 28.6.10.1.1 One Z or No Z for the String

If there is a non-null constant z value for the entire string then the z value is given by a **z** block:

```
<z> z_value </z>
```

where *z\_value* is the constant z value for the entire string.

And when there is a constant z, or no z, for the string, then only the (x,y) coordinates are required for each vertex and these are given in a **data\_2d** block. See [28.3.12 data\\_2d](#)

### 28.6.10.1.2 Varying Z Values along the String

If the z value can vary for different vertices along the string then the (x,y,z) values must be given for each vertex and the data is then written out as a **data\_3d** block. See [28.3.13 data\\_3d](#).

## 28.6.10.2 Geometry of the Horizontal Segments

If the segments are straight lines only then that is the default and no further information is required.

If the segments are only straight lines and arcs, then the **radius\_data** and **major\_data** blocks are used to define a **radius** and **bulge\_flag** data for each segment of the super string. See [28.6.10.2.1 Only Straights and Arcs for Segments](#).

If any of the segments are transitions or offset transitions then **geometry\_data** must be used for each segment. **geometry\_data** can represent a straight, arc, transition or offset transition. See [28.6.10.2.2 Straights, Arcs and Transitions for Segments](#).

### 28.6.10.2.1 Only Straights and Arcs for Segments

If there are only straight and arc segments for the string, then for either **data\_2d** or **data\_3d**, it is possible to add a radius and major/minor arc flag for each segment of the super string using the **radius\_data** and **major\_data** blocks respectively. See [28.3.14 radius\\_data and major\\_data](#).

### 28.6.10.2.2 Straights, Arcs and Transitions for Segments

When some of the segments are transitions or offset transitions, then the **geometry\_data** block **must** be used to give the geometry for each segments.

Either a **data\_2d** or **data\_3d** block defines the coordinates for the vertices and the **geometry\_data** block defines for each segment whether the segment is a straight, an arc or a transition or offset transition.

The definition of the **geometry\_data** block is

```
<geometry_data>
  info_for_segment_1_block
  info_for_segment_2_block
  ...
  info_for_segment_m_block
</geometry_data>
```

where

*info\_for\_segment\_i\_block* is the information defining the i'th segment as either a straight, an arc or an offset transition (offset transition or transition), and

$m = n-1$  for an open string or  $m = n$  for a closed string.

For the definition of *info\_for\_segment\_i\_block* see:

[28.6.10.2.2.1 Straight](#)

[28.6.10.2.2.2 Arc](#)

[28.6.10.2.2.3 Offset Transitions](#)

### 28.6.10.2.2.1 Straight

No parameters are needed for defining a straight segment. The *straight* block is simply:

```
<straight> </straight>
```

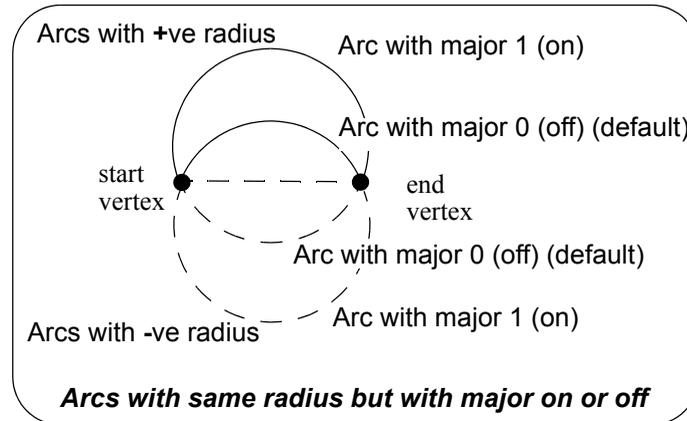
or simply

```
<straight/>
```

### 28.6.10.2.2.2 Arc

There are four possibilities for an arc of a given radius placed between two vertices.

We use *positive* and *negative* radius, and a flag *major* which can be set to 1 (on) or off (0) to differentiate between the four possibilities.



The *arc* block is:

```
<arc>
```

```
<radius> radius_for_segment</radius>
```

```
<major> major_flag_for_segment</major>
```

```
</arc>
```

where

*radius\_for\_segment* is the radius for the segment and

*major\_flag\_for\_segment* is 1 if the arc is a major arc and 0 if it is a minor arc.

### 28.6.10.2.2.3 Offset Transitions

An **offset transition** is a fixed perpendicular offset (**offset\_real**) of a base transition where the base transition is a Euler spiral (or a certain approximation to it) or some other specially defined curve. The base transition has a start point where the absolute radius of the curve is infinity and then has a continuously decreasing absolute radius as you continue along the curve (this may be in a forward or reverse direction).

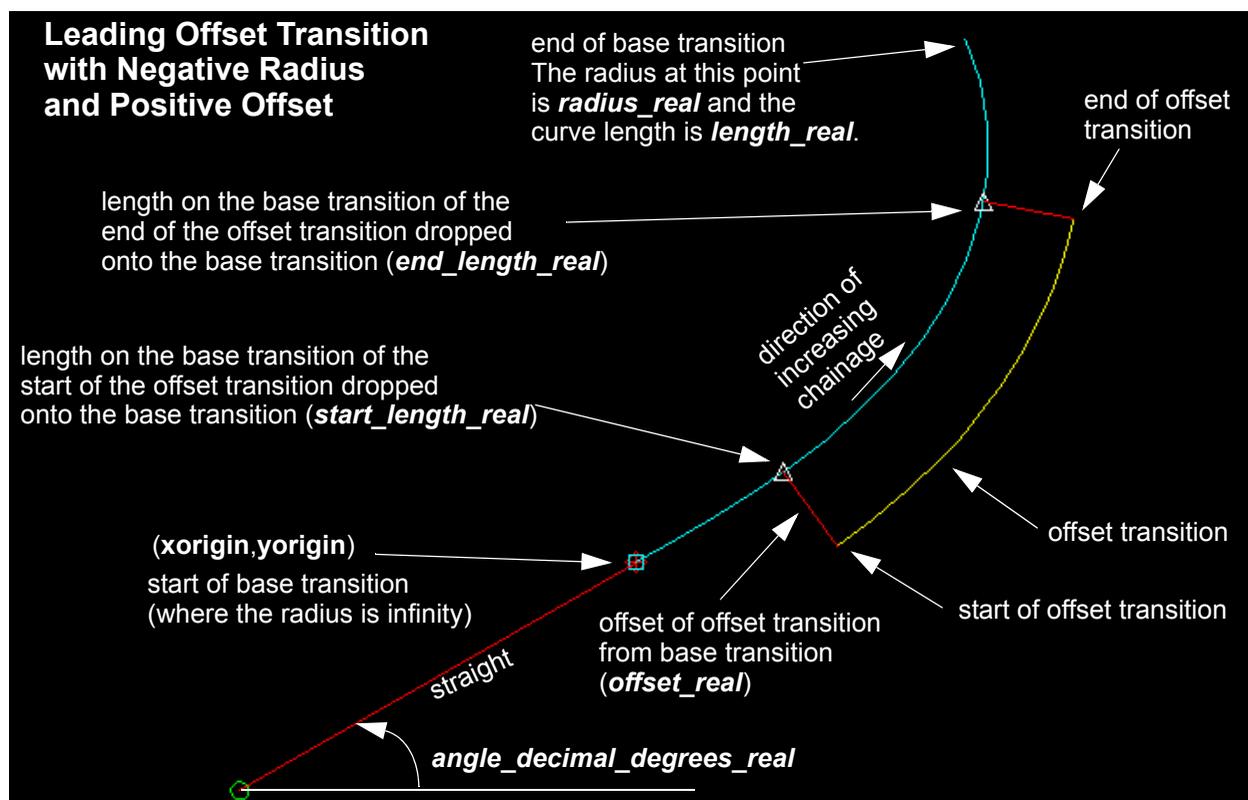
The **base transition** is defined by giving its starting point (**xorigin**, **yorigin**) where the radius is infinity and the angle of the tangential line at the start point is **angle\_decimal\_degrees\_real** and the fact that the radius **radius\_real** occurs at a given curve length **length\_real** along the base transition.

The **offset transition** is a fixed offset (**offset\_real**) from the base transition and goes from a start point that is specified by giving the length on the base transition where the start point drops perpendicularly onto the base transition (**start\_length\_real**) and to the end point that is specified by length on the base transition where the end point drops perpendicularly onto the base transition (**end\_length\_real**). The offset can be positive or negative.

If you are travelling along the curve in a forward direction (increasing chainage) then the base transition is said to be a **leading transition** if the absolute radius decreases as you go along the curve, and a **trailing transition** if the absolute radius increases.

The **end radius** can be **positive** or **negative**.

If you are travelling along the curve in a forward direction then for a leading transition, if the **end radius** is **positive** then the curve curls to the right, and for a negative end radius, the curve curls to the left.



The **curve** block covers both spiral and non-spiral transitions with a zero or non zero offset.

The **curve** block is:

```
<curve>
  <curve_type> curve_type_text</curve_type>
  <leading> leading_logical_text</leading>
```

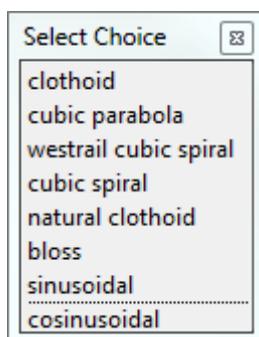
```

<xorigin> xorigin_real</xorigin>
<yorigin> yorigin_real</yorigin>
<radius> radius_real</radius>
<length> length_real</length>
<start> start_length_real</start>
<end> end_length_real</end>
<angle> angle_decimal_degrees_real</angle>
<offset> offset_real</offset>
<mvalue> mvalue_real</mvalue>
</curve

```

where

***curve\_type\_text*** is the type of base transition.



For more information on the choices, see [28.3.15 Available Transition Types](#).

***leading\_logical\_text*** is *true* if it is a **leading** base transition or *false* if it is a trailing base transition.

***(xorigin, yorigin)*** is the origin of the base transition. That is, where the radius is infinity.

***radius\_real*** is the radius at the end of the base transition. The radius is positive if the curve goes to the right when travelling in decreasing absolute radius.

***length\_real*** is the curve length and the end of the base transition and the radius is *radius\_real*.

***start\_length\_real*** is the curve length on the base transition where the start of the offset transition drops perpendicularly onto the base transition.

***end\_length\_real*** is the curve length on the base transition where the end of the offset transition drops perpendicularly onto the base transition.

***angle\_decimal\_degrees\_real*** is the angle of the tangent of the base transition at the origin of the base transition. It is measured in decimal degrees in a counter clockwise direction from the positive x-axis.

***offset\_real*** is the perpendicular offset distance of the transition from the base transition. For a leading transition, a positive value offsets the base transition to the right and a negative value offsets it to the left as you travel in a forward direction.

***mvalue\_real*** - if the transition is a cubic parabola then *mvalue\_real* is the mvalue for the cubic parabola, otherwise, *mvalue\_real* is zero.

For example, for a string with four segments

```
<geometry_data>
```

```
<arc>
  <radius>-222.77841769</radius>
  <major>0</major>
</arc>
<curve>
  <type>clothoid</type>
  <leading>>false</leading>
  <xorigin>114.78632204</xorigin>
  <yorigin>22.22840069</yorigin>
  <radius>222.77841769</radius>
  <length>194.18990415</length>
  <start>50.95749554</start>
  <end>194.18990415</end>
  <angle>174.01773651</angle>
  <offset>0</offset>
  <mvalue>0</mvalue>
</curve>
<arc>
  <radius>-848.96871636</radius>
  <major>0</major>
</arc>
<straight/>
</geometry_data>
```

### 28.6.10.3 Colour

There can be one colour for the entire super string which is given by the **<colour>** keyword block in the **string\_header\_block**, or the colour varies for each segment of the super string and is then specified in a **<colour\_data>** block.

The order of the entries in the **<colour\_data>** block must match the order of the segments in the super string. So there is exactly one entry for each segment.

If all the segment are the string colour, then simply omit the **<colour\_data>** block.

For a super string with **n** vertices

```
<colour_data>
  colour_text_for_segment_1
  colour_text_for_segment_2
  ...
  colour_text_segment_m
</colour_data>
```

where

**colour\_text\_segment\_i** is the colour name or colour number for the *i*'th segment OR is **no\_colour** when no special colour has been set for the segment and the string colour is then used for that segment. If the name includes spaces then it must be enclosed in **"**, and

**m = n-1** for an open string or **m = n** for a closed string.

For example for a string with four segments

```
<colour_data>
  "off yellow" magenta no_colour no_colour
</colour_data>    <leading>false</leading>
```

## 28.6.10.4 String, Vertex and Segment Attributes

The super string can have attributes for the entire string (string attributes) but can also have attributes for each vertex (vertex attributes) and attributes for each segment (segment attributes).

See

[28.6.10.4.1 String Attributes](#)

[28.6.10.4.2 Vertex Attributes](#)

[28.6.10.4.3 Segment Attributes](#)

### 28.6.10.4.1 String Attributes

There can be attributes for the entire string. They are part of the String Header Block and are described in [28.6.3 String Header Block](#).

For example

```
<string_super>
  <name>Line 1</name>
  <chainage>0</chainage>
  <breakline>line</breakline>
  <colour>cyan</colour>
  <style>1</style>
  <attributes>
    <text>
      <name>Street</name>
      <value>Weemala Road</value>
    </text>
  </attributes>
  <time_created>2015-05-11T09:08:06Z</time_created>
  <time_updated>2015-05-11T11:59:29Z</time_updated>
  ...
```

### 28.6.10.4.2 Vertex Attributes

Each vertex can have one or more user defined attributes.

For a super string with *n* vertices

```
<vertex_attribute_data>  
  vertex_attributes_for_vertex_1_block  
  vertex_attributes_for_vertex_2_block  
  ...  
  vertex_attributes_for_vertex_n_block  
</vertex_attribute_data>
```

where

***vertex\_attributes\_for\_vertex\_j\_block*** is the *attribute\_block* for vertex *j*. The *attribute\_block* is defined in [28.4 Attributes](#).

For example, for a string with four vertices

```
<vertex_attribute_data>  
  <attributes>  
    <real>  
      <name>Flow</name>  
      <value>27.4</value>  
    </real>  
  </attributes>  
</attributes/>  
</attributes/>  
</attributes/>  
</vertex_attribute_data>
```

### 28.6.10.4.3 Segment Attributes

Each segment can have one or more user defined attributes.

For a super string with  $n$  vertices

```
<segment_attribute_data>
  segment_attributes_for_segment_1_block
  segment_attributes_for_segment_2_block
  ...
  segment_attributes_for_segment_m_block
</segment_attribute_data>
```

where

***segment\_attributes\_for\_segment\_j\_block*** is an *attribute\_block* for segment  $j$ . The *attribute\_block* is defined in [28.4 Attributes](#), and

$m = n-1$  for an open string or  $m = n$  for a closed string.

For example, for an open string with four vertices

```
<segment_attribute_data>
  <attributes>
    <real>
      <name>Material</name>
      <value>clay</value>
    </real>
  </attributes>
</attributes/>
</attributes/>
</attributes/>
</segment_attribute_data>
```

### 28.6.10.5 Vertex Id's (Point Id's)

Each vertex can have a vertex id (point id).

This is not the number position of the vertex in the string but is a separate id which is usually different for every vertex in every string.

The **vertex id** can be alphanumeric and include spaces.

The definition is:

For a super string with **n** vertices

```
<point_data>
  point_id_text_for_vertex_1
  point_id_text_for_vertex_2
  ...
  point_id_text_for_vertex_n
</point_data>
```

where

*point\_id\_text\_for\_vertex\_i* is the point id of the *i*'th vertex.

$m = n-1$  for an open string or  $m = n$  for a closed string.

*point\_id\_text\_for\_vertex\_i* is the actual text enclosed in "", even if the text does not include spaces. If the point id has not been set for a vertex, the value should be included as "".

For example "Point 1" or "Point2" or "".

If the **point\_data** block does not exist then there are no Vertex Ids.

For example, for a string with 4 vertices

```
<point_data>
  "Point 1" "Point2" "" ""
</point_data>
```

### 28.6.10.6 Symbols at Vertices

There can be no symbols at all, or the same symbol for every vertex in the using the ***symbol\_value*** block or the symbol varies for each vertex of the super string using the ***symbol\_data*** block.

If a symbol does not have a colour, or there is no colour in the symbol definition, then it uses the string colour.

The definitions are:

**<symbol\_value>**

*symbol\_properties\_block*

**</symbol\_value>**

where

***symbol\_properties\_block*** is the description for the symbol to be used at every vertex of the super string, and

OR

For a super string with **n** vertices

**<symbol\_data>**

*symbol\_properties\_for\_vertex\_1\_block*

*symbol\_properties\_for\_vertex\_2\_block*

...

*symbol\_properties\_for\_vertex\_n\_block*

**</symbol\_data>**

where

***symbol\_properties\_for\_vertex\_i\_block*** is the description for the symbol at vertex *i*.

The format of ***symbol\_properties\_block*** and ***symbol\_properties\_for\_vertex\_i\_block*** is:

**<properties>**

**<style>** *symbol\_name\_text* **</style>**

**<colour>** *symbol\_colour\_name\_text* **</colour>**

**<size>** *symbol\_size\_real* **</size>**

**<rotation>** *angle\_dec\_deg\_real* **</rotation>**

**<offset\_x>** *symbol\_offset\_x\_real* **</offset\_y>**

**<offset\_y>** *symbol\_offset\_y\_real* **</offset\_y>**

**</properties>**

where

***symbol\_name\_text*** is the name of the symbol.

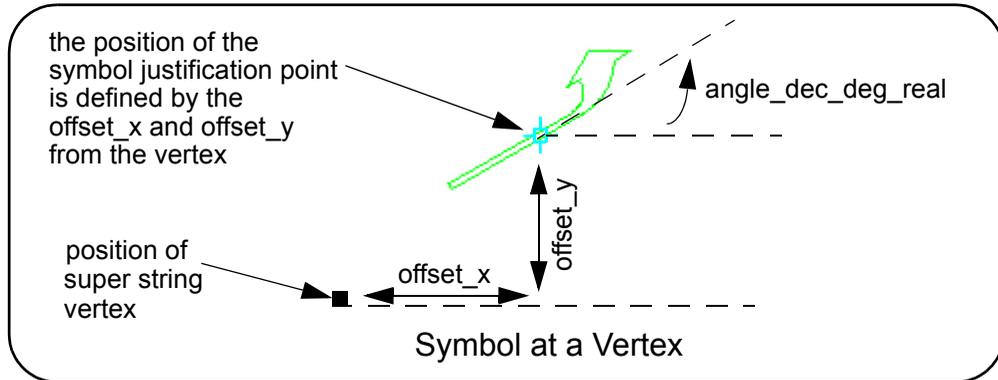
***symbol\_colour\_name*** is the colour of the symbol is there is no colours in the symbol definition. If the ***colour*** block is missing and there is no colours in the symbol definition then the string colour is used. For the syntax of the ***colour*** block, see [28.3.2 Colour](#).

***symbol\_size\_real*** is the size of the symbol in the units of the symbol.

***angle\_dec\_deg\_real*** is the angle of the symbol. The value is in decimal degrees and is measured in a counter clockwise direction from the positive x-axis.

***offset\_x\_real*** is x distance to offset the symbol from the super string vertex.

**offset\_y\_real** is the y distance to offset the symbol from the super string.



### 28.6.10.7 Tinability

#### Tinability

For a *super string*, the concept of breakline has been extended to a property called **tinable** which can be set independently for each vertex and each segment of the super string.

If a vertex is tinable, then the vertex is used in triangulations. If the vertex is not tinable, then the vertex is ignored when triangulating.

If a segment is tinable, then the segment is used as a side of a triangle during triangulation. This may not be possible if there are *crossing* tinable segments.

*Vertex tinability* is given by the **vertex\_tinable\_data** block where for a string of  $n$  vertices,

```
<vertex_tinable_data>
  tinable_flag_for_vertex_1
  tinable_flag_for_vertex_2
  ...
  tinable_flag_for_vertex_n
</vertex_tinable_data>
```

where

**tinable\_flag\_for\_vertex\_i** for the  $i$ 'th vertex is 1 or t if the vertex is tinable, or 0 or f if the vertex is not tinable.

*Segment tinability* is given by the **segment\_tinable\_data** block where

```
<segment_tinable_data>
  tinable_flag_for_segment_1
  tinable_flag_for_segment_2
  ...
  tinable_flag_for_segment_m
</segment_tinable_data>
```

where

**tinable\_flag\_for\_segment\_i** for the  $i$ 'th segment is 1 or t if the segment is tinable, or 0 or f if the segment is not tinable, and

$m = n-1$  for an open string or  $m = n$  for a closed string.

For example, for a open string with four vertices

```
<vertex_tinable_data>
  t t f t
</vertex_tinable_data>
<segment_tinable_data>
  f t t
</segment_tinable_data>
```

**Note** that even if a segment is set to tinable, it can only be used in a triangulation if both its end vertices are also tinable.

### 28.6.10.8 Round or Box (Culvert) Pipes

All the super string segments can be round or all box (culvert). That is, some can't be round and others box.

There is also a justification that applies to all round pipes or box pipes in the string.

See

[28.6.10.8.1 Pipe Diameters](#)

[28.6.10.8.2 Culvert Dimensions](#)

[28.6.10.8.3 Justification for Round or Culvert Pipes](#)

#### 28.6.10.8.1 Pipe Diameters

There can be one pipe diameter value for the entire super string using the ***pipe\_value*** block or the pipe diameter varies for each segment of the super string using the ***pipe\_data*** block.

The definitions are:

```
<pipe_value> pipe_diameter_real </pipe_value>
```

where *pipe\_diameter\_real* is the diameter for **every** segment of the string.

OR

For a super string with **n** vertices

```
<pipe_data>
```

```
  <properties>
```

```
    <diameter> pipe_diameter_for_segment_1 </diameter>
```

```
  </properties>
```

```
  <properties>
```

```
    <diameter> pipe_diameter_for_segment_2 </diameter>
```

```
  </properties>
```

```
  ...
```

```
  <properties>
```

```
    <diameter> pipe_diameter_for_segment_m </diameter>
```

```
  </properties>
```

```
</pipe_data>
```

where

*pipe\_diameter\_for\_segment\_i* is the pipe diameter for the *i*'th segment, and

$m = n-1$  for an open string or  $m = n$  for a closed string.

#### 28.6.10.8.2 Culvert Dimensions

There can be one culvert width and height for the entire super string using the ***culvert\_value*** block or the culvert width and height vary for each segment of the super string using the ***culvert\_data*** block.

The definitions are:

```
<culvert_value>
```

```
  <width> pipe_width_real </width>
```

```

    <height> pipe_height_real </height>
  </culvert_value>

```

where *pipe\_width\_real* is the width and *pipe\_height\_real* is the height for every segment of the string.

OR

For a super string with *n* vertices

```

<culvert_data>
  <properties>
    <width> pipe_width_for_segment_1 </width>
    <height> pipe_height_for_segment_1 </height>
  </properties>
  <properties>
    <width> pipe_width_for_segment_2 </width>
    <height> pipe_height_for_segment_2 </height>
  </properties>
  ...
  <properties>
    <width> pipe_width_for_segment_m </width>
    <height> pipe_height_for_segment_m </height>
  </properties>
</culvert_data>

```

where

*pipe\_width\_for\_segment\_i* is the width and *pipe\_height\_for\_segment\_i* is the height for the *i*'th segment and

$m = n - 1$  for an open string or  $m = n$  for a closed string.

### 28.6.10.8.3 Justification for Round or Culvert Pipes

There can be only one justification for all the round or culvert pipe segments in the super string.

The definition is:

```

<justify> pipe_justification_text </justify>

```

where

*pipe\_justification\_text* is the justification for the entire pipe and can have the values **centre**, **top**, **obvert**, **bottom** or **invert**.

If the *justify* block is missing then the round pipe or culvert pipe is **centre** justified.

## 28.6.10.9 Vertex Text and Vertex Annotation

See

[28.6.10.9.1 Vertex Text](#)

[28.6.10.9.2 Vertex Annotation](#)

### 28.6.10.9.1 Vertex Text

There can be not text at each vertex, the same piece of text for every vertex in the super string or a different text for each vertex of the super string.

Note: How the vertex text is drawn is specified by the vertex annotation. See [28.6.10.9.2 Vertex Annotation](#).

If there is a constant text value for each vertex in the string, then the text value is given by a **vertex\_text\_value** block:

```
<vertex_text_value> text_value_text </vertex_text_value>
```

where *text\_value\_text* is the constant text value for each vertex in the string.

For example, for a string of 5 vertices

```
<vertex_text_value>Constant text</vertex_text_value>
```

If there is a different text value for each vertex in the string, then the value of the text for each vertex is given in a **vertex\_text\_data** block.

```
<vertex_text_data>
```

```
<p> text_value_for_vertex_1</p>
```

```
<p> text_value_for_vertex_2</p>
```

```
...
```

```
<p> text_value_for_vertex_n</p>
```

```
</vertex_text_data>
```

where *text\_value\_for\_vertex\_i* is the vertex text for the *i*'th vertex.

For example, for a string of four vertices

```
<vertex_text_data>
<p>First vertex</p>
<p>Second vertex</p>
<p/>
<p/>
</vertex_text_data>
```

### 28.6.10.9.2 Vertex Annotation

How the vertex text is drawn at each vertex is specified by the vertex annotation.

There can be no vertex annotations at all, or the same vertex annotation is used for every vertex in the string using the ***vertex\_annotation\_value*** block, or the vertex annotation varies for each vertex of the super string using the ***vertex\_annotation\_data*** block.

Note that in vertex annotations, the size of the text for all vertices must be either world size or all paper size or all screen size. That is, world size, paper size and screen size can not be mixed. The first one found is used for all vertices.

The definitions are:

```
<vertex_annotate_value>
  vertex_annotation_information
</vertex_annotate_value>
```

where

*vertex\_annotation\_information* is the annotation to be used for drawing the text at every vertex of the super string. For the definition of *vertex\_annotation\_information* see [28.6.4.1 Vertex Annotation Information](#).

OR

For a super string with *n* vertices

```
<vertex_annotation_data>
  annotation_for_vertex_1_block
  annotation_for_vertex_2_block
  ...
  annotation_for_vertex_n_block
</vertex_annotation_data>
```

where

***annotation\_for\_vertex\_i\_block*** is the description for the vertex annotation for vertex *i*.

The format of the ***annotation\_for\_vertex\_i\_block*** is:

```
<properties>
  vertex_annotation_information
</properties>
```

where

*vertex\_annotation\_information* is the annotation for drawing the text at the vertex. For the definition of *vertex\_annotation\_information* see [28.6.4.1 Vertex Annotation Information](#).

## 28.6.10.10 Segment Text and Segment Annotation

See

[28.6.10.10.1 Segment Text](#)

[28.6.10.10.2 Segment Annotation](#)

### 28.6.10.10.1 Segment Text

There can be no text on each segment, the same piece of text for every segment in the super string or a different text for each segment of the super string.

Note: How the segment text is drawn is specified by the segment annotation. See [28.6.10.10.2 Segment Annotation](#).

If there is a constant text value for each segment in the string, then the text value is given by a **segment\_text\_value** block:

```
<segment_text_value> text_value_text </segment_text_value>
```

where *text\_value\_text* is the constant text value for each segment in the string.

For example, for a string of 5 vertices

```
<segment_text_value>Constant text</segment_text_value>
```

If there is a different text value for each segment in the string, then the value of the text for each segment is given in a **segment\_text\_data** block.

```
<segment_text_data>
  <p>text_value_for_segment_1</p>
  <p>text_value_for_segment_2</p>
  ...
  <p> text_value_for_segment_m</p>
</segment_text_data>
```

where

*text\_value\_for\_segment\_i* is the segment text for the *i*'th segment, and

$m = n - 1$  for an open string or  $m = n$  for a closed string.

For example, for a string of four segments

```
<segment_text_data>
  <p>First segment</p>
  <p>Second Segment&#xa;Two lines</p>
  <p>seg3</p>
  <p/>
</segment_text_data>
```

### 28.6.10.10.2 Segment Annotation

How the segment text is drawn at each segment is specified by the segment annotation.

There can be no segment annotations at all, or the same segment annotation is used for every segment in the string using the ***segment\_annotation\_value*** block, or the segment annotation varies for each segment of the super string using the ***segment\_annotation\_data*** block.

Note that in segment annotations, the size of the text for all segments must be either world size or all paper size or all screen size. That is, world size, paper size and screen size can not be mixed. The first one found is used for all segments.

The definitions are:

```
<segment_annotate_value>
  segment_annotation_information
</segment_annotate_value>
```

where

*segment\_annotation\_information* is the annotation to be used for drawing the text at every segment of the super string. For the definition of *segment\_annotation\_information* see [28.6.4.2 Segment Annotation Information](#).

OR

For a super string with *n* vertices

```
<segment_annotation_data>
  annotation_for_segment_1_block
  annotation_for_segment_2_block
  ...
  annotation_for_segment_m_block
</segment_annotation_data>
```

where

*annotation\_for\_segment\_i\_block* is the description for the annotation at segment *i*, and.  
 $m = n - 1$  for an open string or  $m = n$  for a closed string.

The format of the ***annotation\_for\_segment\_i\_block*** is:

```
<properties>
  segment_annotation_information
</properties>
```

where

*segment\_annotation\_information* is the annotation for drawing the text at the segment. For the definition of *segment\_annotation\_information* see [28.6.4.2 Segment Annotation Information](#).

Continue to the next section [28.6.11 Super Alignment String](#) or return to [28.6 Elements Contained in Models](#) or [28 12d XML File Format](#).

## 28.6.11 Super Alignment String

Many software packages only allow *alignment* strings to be use the intersection point method (IP's) to construct the horizontal and vertical geometry. The IP definition is actually a *constructive* definition and the tangents points and segments between the tangent points (lines, arcs, transitions etc.) are calculated from the IP definition.

However for the **12d Model super alignment**, the **horizontal** and **vertical geometry** are still defined separately and with construction definitions **but** the construction definition can be much more complex than just IP's. For example, an arc could be defined as being tangential to two offset elements, or constrained to go through a given point.

If the horizontal construction methods are consistent then the horizontal geometry can be solved, and the horizontal geometry expressed in terms of consecutive segments (lines, arcs, transitions) that are easily understood and drawn.

Similarly if the vertical construction methods are consistent then the vertical geometry can be solved, and the vertical geometry expressed in terms of consecutive segments (lines, arcs, parabolas) that are easily understood and drawn.

For the *super alignment* **both** the **construction methods** (the **parts**) and the resulting **vertices** and **segments** (lines, arcs, transitions etc.) that make up the horizontal and vertical geometry (the **data**) are written out to the 12d XML file.

For most applications such as uploading to survey data collectors or machine control devices, only the **horizontal data** and the **vertical data** are required, not the *construction* methods (i.e. not the **horizontal** and **vertical parts**). So when reading the 12d XML of a *super alignment*, only the **horizontal** and **vertical data** needs to be read in and the constructive methods (the **horizontal** and **vertical parts**) can be skipped over.

Consequently only the **horizontal data** and the **vertical data** is full documented for the super alignment.

However to allow **12dXML** to be easily written out by software packages that can only support HIP and VIP methods, there are special flags to denote these cases and the `horizontal_parts` and `vertical_parts` are fully defined for these special cases.

### Special Cases for HIP Method Only and VIP Method Only

To allow 12dXML to be easily written out by software packages that can only support HIP and VIP methods, there are special flags to denote these cases and the `horizontal_parts` and `vertical_parts` are fully defined for these special cases. This also means that such a software package can easily read in from 12dXML any super alignments that use HIP and VIP methods only.

- (a) If the **horizontal geometry** of the super alignment **only uses the HIP method** and hence only has Horizontal IP's with curves and transitions on them, then the HIP definition can be easily read in from the `horizontal_parts`. To alert any software reading 12dXML reader of this special case, there is a special flag `horizontal_ips_only` which is then set to **true**. Other wise it is false.

This special case for `horizontal_parts` is fully documented in [28.6.11.2 Horizontal\\_Parts When Geometry is Defined by IP Method Only](#).

- (b) If the **vertical geometry** of the super alignment **only uses the VIP method** and hence only has Vertical IP's with parabolic curves and arcs on them, then the VIP definition can be easily read in from the `vertical_parts`. To alert any software reading 12dXML reader of this special case, there is a special flag `vertical_ips_only` which is then set to **true**. Other wise it is false.

This special case for `vertical_parts` is fully documented in [28.6.11.5 Vertical\\_parts When VG is Defined by IP Method Only](#).

## Notes

1. Just using the horizontal and vertical data is valid **as long as the super alignment geometry is consistent** (and solves) and the horizontal and vertical parts can be created. There are the flags **valid\_horizontal** and **valid\_vertical** in the 12d XML of the super alignment and they are set to **true** if the horizontal and vertical geometry is consistent and solves.
2. Segments meeting at a common vertex do not have to be tangential although for most road and rail centre lines, they should be.
3. When **12d Model** reads in a 12d XML file and there is only **horizontal\_parts** and no **horizontal\_data** then if possible, **12d Model** generates the **horizontal\_data** from the horizontal parts.

This is very useful if you are creating a 12d XML file for a super alignment string that only uses HIP methods as it is fairly simple to create the **horizontal\_parts** for such a string and that is fully documented in [28.6.11.2 Horizontal\\_Parts When Geometry is Defined by IP Method Only](#). For this case the flag **horizontal\_ips\_only** should be set to **true**.

4. When **12d Model** reads in a 12d XML file and there is only **vertical\_parts** and no **vertical\_data** then if possible, **12d Model** generates the **vertical\_data** from the vertical parts.

This is very useful if you are creating a 12d XML file for a super alignment string that only uses VIP methods as it is fairly simple to create the **vertical\_parts** for such a string and that is fully documented in [28.6.11.5 Vertical\\_parts When VG is Defined by IP Method Only](#). For this case the flag **vertical\_ips\_only** should be set to **true**.

The 12d XML definition of the super alignment string is:

```
<string_super_alignment>
  string_header_block
  drawables_block
  spiral_type_block
  closed_block
  valid_horizontal_block
  valid_vertical_block
  synch_vertical_block
  label_style_block
  horizontal_ips_only_block
  vertical_ips_only_block
  horizontal_parts_block
  horizontal_data_block
  vertical_parts_block
  vertical_data_block
  geometry_modifiers_block
</string_super_alignment>
```

where

**string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

**drawables\_block**

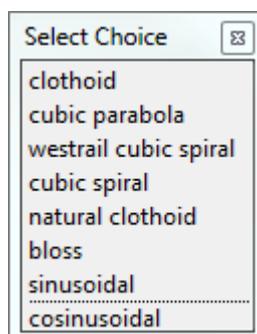
the drawables block contains information on how the super alignment is labelled.

This block is not documented.

**spiral\_type\_block**

```
<spiral_type> transition_type_text </spiral_type>
```

where *transition\_type\_text* is the default transition type use in the super alignment and is one of



For more information on the choices, see [28.3.15 Available Transition Types](#).

**closed\_block**

```
<closed> closed_text_logical </closed>
```

where *closed\_text\_logical* is **true** if the super alignment string is closed and **false** if the

super alignment string is open.

#### **valid\_horizontal\_block**

**<valid\_horizontal>** *valid\_horizontal\_text\_logical* **</valid\_horizontal>**

where *valid\_horizontal\_text\_logical* is **true** if the super alignment string horizontal geometry solves and **false** if the horizontal geometry does not solve.

If the horizontal geometry does not solve then the *horizontal\_data* may be rubbish.

#### **valid\_vertical\_block**

**<valid\_vertical>** *valid\_vertical\_text\_logical* **</valid\_vertical>**

where *valid\_vertical\_text\_logical* is **true** if the super alignment string vertical geometry solves and **false** if the vertical geometry does not solve.

If the vertical geometry does not solve then the *vertical\_data* may be rubbish.

#### **synch\_vertical\_block**

**<synch\_vertical>** *synch\_vertical\_text\_logical* **</synch\_vertical>**

where *synch\_vertical\_text\_logical* is **true** if the super alignment vertical geometry is to be synchronized to the horizontal geometry whenever the horizontal geometry is modified.

This is an internal **12d Model** flag.

#### **label\_style\_block**

**<label\_style>** *label\_style\_text* **</label\_style>**

where *label\_style\_text* is the name of the super alignment label style used for drawing the super alignment.

#### **horizontal\_ips\_only\_block**

**<horizontal\_ips\_only>** *horizontal\_ips\_only\_text\_logical* **</horizontal\_ips\_only>**

where *horizontal\_ips\_only\_text\_logical* is **true** if the horizontal geometry of the super alignment consists of HIP methods only, and **false** if the horizontal geometry does not consist of HIP methods only.

#### **vertical\_ips\_only\_block**

**<vertical\_ips\_only>** *vertical\_ips\_only\_text\_logical* **</vertical\_ips\_only>**

where *vertical\_ips\_only\_text\_logical* is **true** if the vertical geometry of the super alignment consists of VIP methods only, and **false** if the vertical geometry does not consist of HIP methods only.

#### **horizontal\_parts\_block**

the *horizontal\_parts* block contains the **methods** to construct the super alignment horizontal geometry. For example float (fillet) an arc of a certain radius between two given lines or create a transition (spiral or non-spiral transition) between a line and an arc.

The parts that make up the horizontal geometry are defined in chainage order from the start to the end of the super alignment.

If the horizontal construction methods are consistent, then they can be solved to form a plan string made up of lines, arcs and transitions and this is given in the **horizontal\_data** block.

Because the construction methods can be very complex, the *horizontal\_parts* block will only be documented for the case where all the horizontal parts are horizontal intersection points (HIPs) with an arc and leading and trailing transitions. See [28.6.11.2 Horizontal\\_Parts When Geometry is Defined by IP Method Only](#).

#### **horizontal\_data\_block**

the *horizontal\_data* block contains the segments that define the horizontal geometry.

The *horizontal\_data* block needs to be read in.

For the description of the *horizontal\_data* block, see [28.6.11.1 Horizontal Data Block](#).

#### ***vertical\_parts\_block***

the *vertical\_parts* block contains the **methods** to construct the super alignment vertical geometry. For example float (fillet) an arc of a certain radius between two given lines.

The parts that make up the vertical geometry are defined in chainage order from the start to the end of the super alignment.

If the vertical construction methods are consistent, then they can be solved to form a string in (chainage, offset) space made up of lines, arcs and parabolas and this is given in the ***vertical\_data*** block.

Because the construction methods can be very complex, the *vertical\_parts* block will only be documented for the case where all the vertical parts are vertical intersection points (VIPs) with an arc or a parabola on the VIP. See [28.6.11.5 Vertical\\_parts When VG is Defined by IP Method Only](#).

#### ***vertical\_data\_block***

the *vertical\_data* block contains the segments that define the vertical geometry.

The *vertical\_data* block needs to be read in.

For the description of the *vertical\_data* block, see [28.6.11.3 Vertical Data Block](#).

#### ***geometry\_modifiers\_block***

the *geometry\_modifiers\_parts* block contains extra construction information for the super alignment.

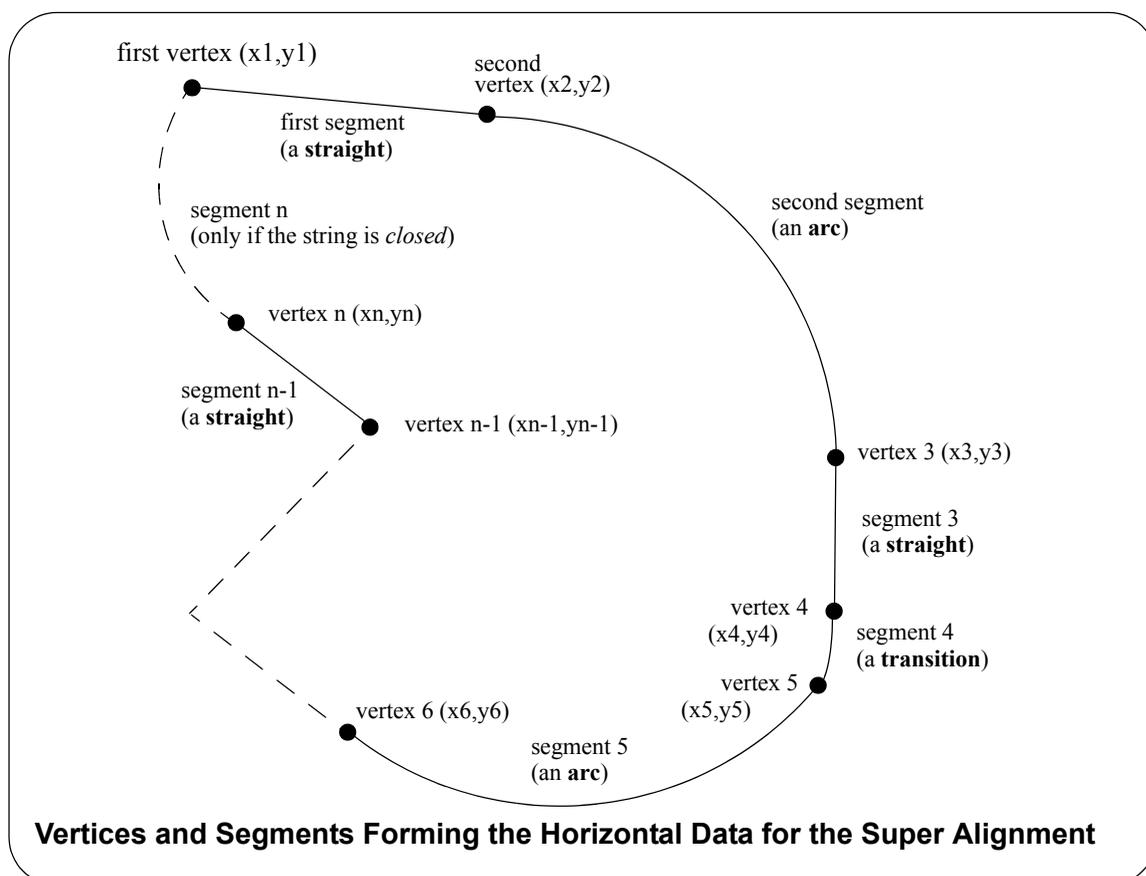
This block is not documented.

### 28.6.11.1 Horizontal Data Block

The *horizontal\_data* block contains the **solved** horizontal geometry of the super alignment.

The *solved horizontal geometry* is made up of a series of (x,y) vertices given in a *data\_2d* block followed by a *geometry\_data* block specifying the geometry of the segments between adjacent vertices. Each segment can be a straight line, an arc, a transition or an offset transition.

If the solved horizontal geometry has **n** vertices, then there will be **n-1 segments** for an **open** super alignment or **n segments** if the super alignment is **closed**.



The format of the *horizontal\_data* block is the same as for the segments of a super string except that the data is only in 2D. Unlike a super string where there is just a z-value at each vertex, the third dimension of the super alignment is given by the *vertical\_data* block (see [28.6.11.3 Vertical Data Block](#)).

The definition of the *horizontal\_data* block is:

```
<horizontal_data>
  string_header_block
  closed_block
  interval_block
  data_2d_block
  geometry_data_block
  blocks_of_info_1
  blocks_of_info_2
  ...

```

*blocks\_of\_info\_n*

</horizontal\_data>

where

***string\_header\_block***

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#). This provides information such as colour for the horizontal data.

***interval\_block***

The *interval\_block* for a super string has a *distance* (a chainage interval) and a *chord\_to\_arc\_real*

where

the ***distance*** to temporarily introduce extra vertices into the string at the given chainage distance when the string is in a triangulation to form a tin.

***chord\_arc\_real*** is a real number and is the chord to arc tolerance to use on any arcs in the horizontal data to temporarily insert vertices into the arc when the arc is included in a triangulation to form a tin.

For the syntax of *interval\_block*, see [28.3.6 Interval](#).

***data\_2d\_block***

the *data\_2d* block defines the (x,y) value of the vertices that makes up the horizontal data.

For the definition of the *data\_2d* block, see [28.3.12 data\\_2d](#).

***geometry\_data\_block***

the segments of the horizontal data can be straights, arcs, transitions or offset transitions.

For the definition of the *geometry\_data* block, see [28.6.10.2 Geometry of the Horizontal Segments](#)

***blocks\_of\_info***

extra information for the vertices and/or segments such as colour, attributes, vertex text, vertex uids etc are defined in the same way as for super strings.

### 28.6.11.2 Horizontal\_Parts When Geometry is Defined by IP Method Only

When the horizontal geometry is defined by IP methods only, then the **horizontal\_parts** is fairly straight forward.

When **12d Model** reads in a 12d XML file and there is no **horizontal\_data** section, then **12d Model** will calculate the **horizontal\_parts**. So you are writing a 12d XML with only IP methods for the horizontal geometry then simply leave out the **horizontal\_data** section and **12d Model** will calculate it for you.

For a horizontal geometry is defined by IP methods only, the **horizontal\_parts** definition is:

```
<horizontal_data>
  info_for_HIP_1_block
  info_for_HIP_2_block
  ...
  info_for_HIP_n_block
</horizontal_data>
```

where **info\_for\_HIP\_i\_block** is the information about the successive HIPs in the super alignment and is one of:

- (a) A horizontal intersection point (HIP) with no arc.

This is defined by:

```
<ip>
  <id> part_id_integer </id>
  time_created_block
  time_updated_block
  <x> x_ip_coordinate_real </x>
  <y> y_ip_coordinate_real </y>
</ip>
```

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**x\_ip\_coordinate\_real** is the x coordinates of the HIP.

**y\_ip\_coordinate\_real** is the y coordinates of the HIP.

- (b) A horizontal intersection point (HIP) with an arc of a given radius at the HIP.

This is defined by:

```
<arc>
  <id> part_id_integer </id>
  time_created_block
  time_updated_block
  <r> arc_radius_real </r>
```

```

    <x> x_ip_coordinate_real </x>
    <y> y_ip_coordinate_real </y>
  </arc>

```

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**arc\_radius\_real** is the radius of the arc on the HIP.

**x\_ip\_coordinate\_real** is the x coordinate of the HIP.

**y\_ip\_coordinate\_real** is the y coordinate of the HIP.

- (c) A horizontal intersection point (HIP) with an arc of a given length at the HIP

This is defined by:

**<length>**

```

    <id> part_id_integer </id>
    time_created_block
    time_updated_block
    <|> arc_length_real </|>
    <x> x_ip_coordinate_real </x>
    <y> y_ip_coordinate_real </y>

```

**</length>**

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**arc\_length\_real** is the length of the arc on the HIP.

**x\_ip\_coordinate\_real** is the x coordinate of the HIP.

**y\_ip\_coordinate\_real** is the y coordinate of the HIP.

- (d) A horizontal intersection point (HIP) with an arc and transitions

This is defined by:

**<spiral>**

```

    <id> part_id_integer </id>

```

```

time_created_block
time_updated_block
transition_type_block
<r> arc_radius_real </r>
<l1> leading_transition_length_real </l1>
<l2> trailing_transition_length_real </l2>
<x> x_ip_coordinate_real </x>
<y> y_ip_coordinate_real </y>

```

</spiral>

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

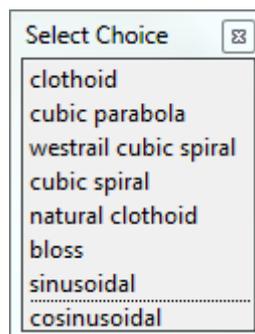
**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**transition\_type\_block**

```
<transition_type> transition_type_text </transition_type>
```

where *transition\_type\_text* is the default transition type use in the super alignment and is one of



This block is optional and if it is missing then the **default transition type** for the super alignment is used.

For more information on the choices, see [28.3.15 Available Transition Types](#).

**arc\_radius\_real** is the radius of the arc on the HIP.

**leading\_transition\_length\_real** is the length of the leading transition on the HIP.

**trailing\_transition\_length\_real** is the length of the trailing transition on the HIP.

**x\_ip\_coordinate\_real** is the x coordinate of the HIP.

**y\_ip\_coordinate\_real** is the y coordinate of the HIP.

**Notes**

1. A <length> block with *arc\_length\_real* equal to zero, or a <spiral> block with the *arc\_radius\_real*, *leading\_transition\_length\_real* and *trailing\_transition\_length\_real* all zero,

will also represent a HIP with no arcs or transitions on it.:

**<length>**

**<id>** *part\_id\_integer* **</id>**

*time\_created\_block*

*time\_updated\_block*

**<l>** 0 **</l>**

**<x>** *x\_ip\_coordinate\_real* **</x>**

**<y>** *y\_ip\_coordinate\_real* **</y>**

**</length>**

OR

**<spiral>**

**<id>** *part\_id\_integer* **</id>**

*time\_created\_block*

*time\_updated\_block*

*transition\_type\_block*

**<r>** 0 **</r>**

**<l1>** 0 **</l1>**

**<l2>** 0 **</l2>**

**<x>** *x\_ip\_coordinate\_real* **</x>**

**<y>** *y\_ip\_coordinate\_real* **</y>**

**</spiral>**

2. If the HIP is the first HIP or the last HIP then no arc or transitions will be drawn even if the relevant parameters are non zero.

As an example of horizontal\_parts with only HIP methods:

```

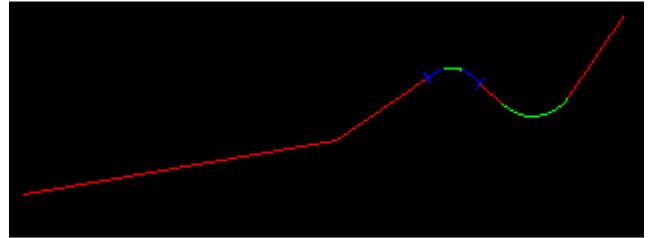
<horizontal_parts>
  <ip>
    <id> 100 </id>
    <x> 42606.66161172 </x>
    <y> 37239.28824481 </y>
  </ip>
  <ip>
    <id> 200 </id>
    <x> 43134.36832349 </x>
    <y> 37330.26705997 </y>
  </ip>
  <spiral>
    <id> 300 </id>
    <r> 50 </r>
    <l1> 30 </l1>
    <l2> 40 </l2>
    <x> 43336.6595 </x>
    <y> 37469.2563 </y>
  </spiral>
  <arc>
    <id> 400 </id>
    <r> 75 </r>
    <x> 43481.15324268 </x>
    <y> 37331.6431906 </y>
  </arc>
  <ip>
    <id> 500 </id>
    <x> 43627.02308964 </x>
    <y> 37544.94343852 </y>
  </ip>
</horizontal_parts>

```

1st HIP  
HIP only

Unique Part id  
incrementing by 100

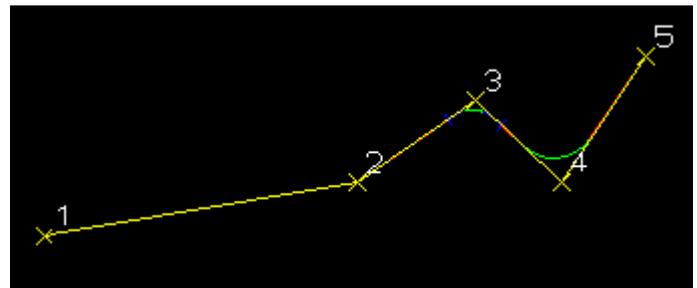
Plan View of Super Alignment



2nd HIP  
HIP only

3rd HIP  
HIP with arc and  
leading and trailing  
transitions

Super Alignment Being Edited



4th HIP  
HIP with arc only

5th HIP  
HIP only

Horizontal Parts with IP Methods Only

### 28.6.11.3 Vertical Data Block

The *vertical\_data* block contains the **solved** vertical geometry of the super alignment.

The *solved vertical geometry* is made up of a series of (chainage,height) vertices given in a *data\_2d* block followed by a *geometry\_data* block specifying the geometry of the segments between adjacent vertices. The segment can be a straight line, a parabola or an arc.

Note that the **chainage** is the chainage of the horizontal geometry defined in the *horizontal\_data* block (see [28.6.11.1 Horizontal Data Block](#)).

If the vertical geometry has **n** vertices, then there will be **n-1 segments** for an **open** super alignment or **n segments** if the super alignment is **closed**.

The format of the *vertical\_data* block is the same as for the segments in a *horizontal\_data* block except that the data is (chainage, height) rather than (x,y) and there is no transitions but a parabola instead.

The definition of the *vertical\_data* block is:

**<vertical\_data>**

*string\_header\_block*

*closed\_block*

*interval\_block*

*data\_2d\_block*

*geometry\_data\_block*

*blocks\_of\_info\_1*

*blocks\_of\_info\_2*

...

*blocks\_of\_info\_n*

**</vertical\_data>**

where

***string\_header\_block***

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#). This provides information such as colour for the vertical data.

***interval\_block***

The *interval\_block* for a super string has a *distance* (a chainage interval) and a *chord\_to\_arc\_real*

where

the **distance** to temporarily introduce extra vertices into the string at the given chainage distance when the string is in a triangulation to form a tin.

**chord\_arc\_real** is a real number and is the chord to arc tolerance to use on any arcs in the vertical data to temporarily insert vertices into the arc when the arc is included in a triangulation to form a tin.

For the syntax of *interval\_block*, see [28.3.6 Interval](#).

***data\_2d\_block***

the *data\_2d* block defines the (chainage,height) value of the vertices that makes up the vertical data.

For the definition of the *data\_2d* block, see [28.3.12 data\\_2d](#) where x is chainage and y is height.

***geometry\_data\_block***

the segments of the vertical data can be straights, arcs or parabolas.

For the definition of the *geometry\_data* block, see [28.6.11.4 Geometry of the Vertical Segments](#)

***blocks\_of\_info***

extra information for the vertices and/or segments such as colour, attributes, vertex text, vertex uids etc are defined in the same way as for super strings.

## 28.6.11.4 Geometry of the Vertical Segments

If the segments are straight lines only then that is the default and no further information is required.

If the segments are only straight lines and arcs, then the **radius\_data** and **major\_data** blocks are used to define a **radius** and **bulge\_flag** data for each segment of the super string. See [28.6.11.4.1 Only Straights and Arcs for Segments](#).

If any of the segments are parabolas then **geometry\_data** must be used for each segment. **geometry\_data** can represent a straight, arc, transition or offset transition. See [28.6.11.4.2 Straights, Arcs and Parabolas for Segments](#).

### 28.6.11.4.1 Only Straights and Arcs for Segments

If there are only straight and arc segments for the string, then for the **data\_2d** it is possible to add a radius and major/minor arc flag for each segment of the super string using the **radius\_data** and **major\_data** blocks respectively. See [28.3.14 radius\\_data and major\\_data](#).

### 28.6.11.4.2 Straights, Arcs and Parabolas for Segments

When some of the segments are parabolas then the **geometry\_data** block **must** be used to give the geometry for each segments.

When the vertical\_data has **n** vertices, then the definition of the **geometry\_data** block is

```
<geometry_data>
  info_for_segment_1_block
  info_for_segment_2_block
  ...
  info_for_segment_m_block
</geometry_data>
```

where

*info\_for\_segment\_i\_block* is the information defining the *i*'th segment as either a straight, an arc or an parabola and  $m = n - 1$  for an open string or  $m = n$  for a closed string.

For the definition of *info\_for\_segment\_i\_block* see:

[28.6.11.4.2.1 Straight](#)

[28.6.11.4.2.2 Arc](#)

[28.6.11.4.2.3 Parabola](#)

### 28.6.11.4.2.1 Straight

No parameters are needed for defining a straight segment. The *straight* block is simply:

```
<straight> </straight>
```

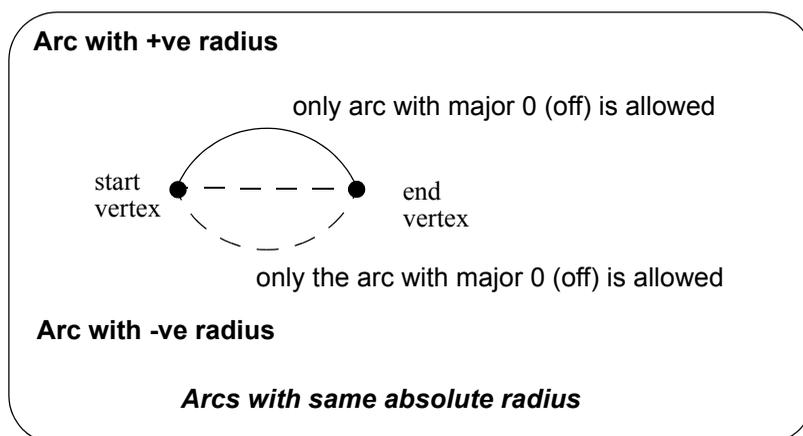
or simply

```
<straight/>
```

### 28.6.11.4.2.2 Arc

Since vertical geometry can't go backwards in chainage value, the majors arcs can not be used and hence there are only possibilities for an arc of a given radius placed between two vertices.

We use *positive* and *negative* radius to differentiate between the four possibilities.



The *arc* block is:

```
<arc>
  <radius> radius_for_segment</radius>
  <major> major_flag_for_segment</major>
</arc>
```

where

*radius\_for\_segment* is the radius for the segment where positive is above the line connecting the vertices.

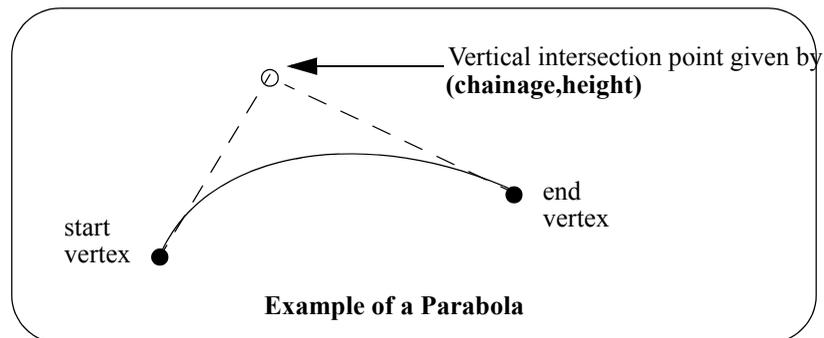
*major\_flag\_for\_segment* is ignored because only minor arcs are allowed.

### 28.6.11.4.2.3 Parabola

There can be a parabola between adjacent vertices. The parabola is defined by giving the coordinates of the vertical intersection point for the parabola

**chainage** chainage of the VIP of the parabola

**height** height of the VIP of the parabola



The *parabola* block is:

**<parabola>**

**<chainage>** *vip\_chainage\_real* **</chainage>**

**<height>** *vip\_height\_real* **</height>**

**</parabola**

where

***vip\_chainage\_real*** is the chainage of the VIP of the parabola

***vip\_height\_real*** is the height of the VIP of the parabola

### 28.6.11.5 Vertical\_parts When VG is Defined by IP Method Only

When the vertical geometry is defined by IP methods only, then the **vertical\_parts** is fairly straight forward.

When **12d Model** reads in a 12d XML file and there is no **vertical\_data** section, then **12d Model** will calculate the **vertical\_parts**. So if you are writing a 12d XML with only VIP methods for the vertical geometry then simply leave out the **vertical\_data** section and **12d Model** will calculate it for you.

For a vertical geometry is defined by VIP methods only, the **vertical\_parts** definition is:

```
<vertical_data>
  info_for_VIP_1_block
  info_for_VIP_2_block
  ...
  info_for_VIP_n_block
</vertical_data>
```

where **info\_for\_VIP\_i\_block** is the information about the successive VIPs in the super alignment and is one of:

- (a) A vertical intersection point (VIP) with no arc or parabola.

This is defined by:

```
<ip>
  <id> part_id_integer </id>
  time_created_block
  time_updated_block
  <x> chainage_ip_coordinate_real </x>
  <y> height_ip_coordinate_real </y>
</ip>
```

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**chainage\_ip\_coordinate\_real** is the chainage of the VIP.

**height\_ip\_coordinate\_real** is the height of the VIP.

- (b) A vertical intersection point (VIP) with an parabola of a given chainage length at the VIP

This is defined by:

```
<length>
  <id> part_id_integer </id>
  time_created_block
  time_updated_block
  <l> parabola_chainage_length_real </l>
```

<x> *chainage\_ip\_coordinate\_real* </x>

<y> *height\_ip\_coordinate\_real* </y>

</length>

where

***part\_id\_integer*** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

***time\_created\_block***

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

***time\_updated\_block***

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

***parabola\_chainage\_length\_real*** is the chainage length of the parabola on the VIP.

***chainage\_ip\_coordinate\_real*** is the chainage of the VIP.

***height\_ip\_coordinate\_real*** is the height of the VIP.

- (c) A vertical intersection point (VIP) with an parabola of a given k value at the VIP

This is defined by:

<kvalue>

<id> *part\_id\_integer* </id>

*time\_created\_block*

*time\_updated\_block*

<k> *parabola\_k\_value\_real* </k>

<x> *chainage\_ip\_coordinate\_real* </x>

<y> *height\_ip\_coordinate\_real* </y>

</kvalue>

where

***part\_id\_integer*** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

***time\_created\_block***

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

***time\_updated\_block***

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

***parabola\_k\_value\_real*** is the k value of the parabola on the VIP.

***chainage\_ip\_coordinate\_real*** is the chainage of the VIP.

***height\_ip\_coordinate\_real*** is the height of the VIP.

- (d) A vertical intersection point (VIP) with an parabola of a given effective radius value at the VIP

This is defined by:

<radius>

<id> *part\_id\_integer* </id>

```

time_created_block
time_updated_block
<r> parabola_effective_radius_value_real </r>
<x> chainage_ip_coordinate_real </x>
<y> height_ip_coordinate_real </y>
</kvalue>

```

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**parabola\_effective\_radius\_value\_real** is the effective radius of the parabola on the VIP.

**chainage\_ip\_coordinate\_real** is the chainage of the VIP.

**height\_ip\_coordinate\_real** is the height of the VIP.

- (e) A vertical intersection point (VIP) with an arc of a given radius at the VIP.

This is defined by:

```

<arc>
  <id> part_id_integer </id>
  time_created_block
  time_updated_block
  <r> arc_radius_real </r>
  <x> chainage_ip_coordinate_real </x>
  <y> height_ip_coordinate_real </y>
</arc>

```

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**arc\_radius\_real** is the radius of the arc on the VIP.

**chainage\_ip\_coordinate\_real** is the chainage of the VIP.

**height\_ip\_coordinate\_real** is the height of the VIP.

- (f) A vertical intersection point (VIP) with an asymmetric parabola defined by the start and end

chainage lengths at that VIP

This is defined by:

**<asymmetric>**

```

<id> part_id_integer </id>
time_created_block
time_updated_block
<l1> parabola_start_chainage_length_real </l1>
<l2> parabola_end_chainage_length_real </l2>
<x> chainage_ip_coordinate_real </x>
<y> height_ip_coordinate_real </y>

```

**</asymmetric>**

where

**part\_id\_integer** is a number that is unique for each horizontal and vertical part and the value is a multiple of 100.

**time\_created\_block**

is the time the super tin was originally created, This is optional. For the syntax see [28.3.7 Time Created](#).

**time\_updated\_block**

is the last time the super tin was last modified, This is optional. For the syntax see [28.3.7 Time Created](#).

**parabola\_start\_chainage\_length\_real** is the start chainage length of the asymmetric parabola on the VIP.

**parabola\_end\_chainage\_length\_real** is the end chainage length of the asymmetric parabola on the VIP.

**chainage\_ip\_coordinate\_real** is the chainage of the VIP.

**height\_ip\_coordinate\_real** is the height of the VIP.

## Notes

1. A <length> block with *arc\_length\_real* equal to zero, or a <spiral> block with the *arc\_radius\_real*, *leading\_transition\_length\_real* and *trailing\_transition\_length\_real* all zero, will also represent a HIP with no arcs or transitions on it.:

**<length>**

```

<id> part_id_integer </id>
time_created_block
time_updated_block
<l> 0 </l>
<x> x_ip_coordinate_real </x>
<y> y_ip_coordinate_real </y>

```

**</length>**

OR

2. If the VIP is the first VIP or the last VIP then no parabola or arc will be drawn even if the relevant parameters are non zero.

As an example of **vertical\_parts** with only VIP methods:

```

<vertical_parts>
  <ip>
    <id> 600 </id>
    <x>-50.8459652 <x>
    <y> 59.79764161 <y>
  </ip>
  <kvalue>
    <id> 700 </id>
    <k> 1.25 </k>
    <x> 38.4627 </x>
    <y> 179.2126 </y>
  </kvalue>
  <length>
    <id> 800 </id>
    <l> 50 </l>
    <x> 172.61694837 </x>
    <y> 154.72967932 </y>
  </length>
  <asymmetric>
    <id> 900 </id>
    <l1> 25 </l1>
    <l2> 75 </l2>
    <x> 270.0182 </x>
    <y> 208.1493 </y>
  </asymmetric>
  <arc>
    <id> 1000 </id>
    <r> 1000 </r>
    <x> 424.2402 </x>
    <y> 196.5637 </y>
  </arc>
  <radius>
    <id> 1100 </id>
    <r> 200 </r>
    <x> 526.7263 </x>
    <y> 201.5302 </y>
  </radius>
  <ip>
    <id> 1200 </id>
    <x> 637.69216273 </x>
    <y> 198.71894484 </y>
  </ip>
</vertical_parts>

```

1st VIP  
VIP only

Unique Part id  
incrementing by 100

2nd VIP  
Parabola defined  
by k value

3rd VIP  
Parabola defined  
by length

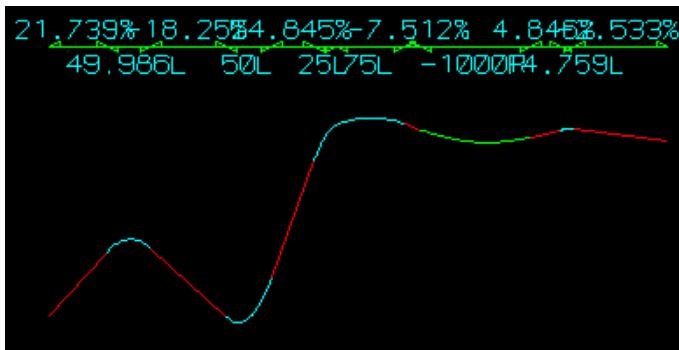
4th VIP  
Asymmetric parabola defined  
by two lengths

5th VIP  
Arc with radius

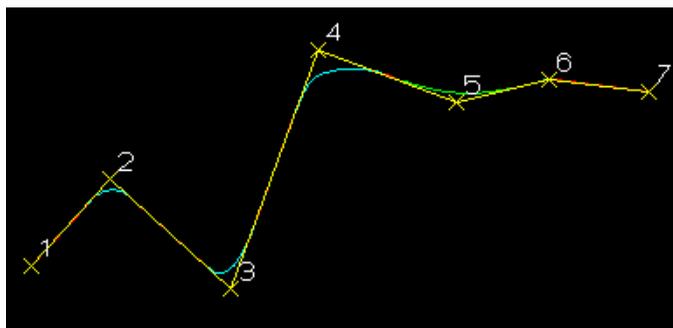
6th VIP  
Parabola defined  
by effective radius

7th VIP  
VIP only

Section View of Super Alignment



Vertical Geometry Being Edited



### Vertical Parts with IP Methods Only

Continue to the next section [28.6.12 Text String](#) or return to [28.6 Elements Contained in Models](#) or [28 12d XML File Format](#).

## 28.6.12 Text String

The format for the **string\_text** element is:

```
<string_text>
  string_header_block
  point_block
  vertex_text_value_block
  vertex_annotate_value_block
</string_text>
```

where

### **string\_header\_block**

the common header block for each string. for the contents and the syntax, see [28.6.3 String Header Block](#).

### **point\_block**

The format of the **point\_block** is:

```
<point> x_real y_real z_real </point>
```

where

(*x\_real,y\_real,z\_real*) is the vertex of the text.

### **vertex\_text\_value\_block**

The text for the text string.

The format of the **vertex\_text\_value\_block** is:

```
<vertex_text_value> characters_of_the_text </vertex_text_value>
```

where

*characters\_of\_the\_text* is the characters of the text with the except of some character that are special characters and are replace by something else.

For example **&** in the text is replaced **&amp;** and a new line is given by **&#xa;**. See [\\_ Characters "<" ">" and "&" and Escaping](#).

### **vertex\_annotate\_block**

These are the setting for displaying text at a vertex.

The format of the **vertex\_annotate\_block** is:

```
<vertex_text_value>
  vertex_annotation_information
</vertex_text_value>
```

where

*vertex\_annotation\_information* is the annotation to be used for drawing the text. For the definition of *vertex\_annotation\_information* see [28.6.4.1 Vertex Annotation Information](#).

For example

```
<string_text>
  <name>text</name>
  <chainage>0</chainage>
  <breakline>line</breakline>
  <colour>yellow</colour>
  <style>1</style>
  <time_created>28-Apr-2015 07:48:35</time_created>
  <time_updated>28-Apr-2015 07:49:33</time_updated>
```

```
<point>1230.93054186 517.0328703 null</point>
<vertex_text_value>First line&#xa;Second line</vertex_text_value>
<vertex_annotate_value>
  <worldsize>20</worldsize>
  <textstyle>Arial</textstyle>
  <angle>45</angle>
  <x_factor>1</x_factor>
  <slant>0</slant>
  <offset>0</offset>
  <raise>0</raise>
  <text_colour>yellow</text_colour>
  <justify>middle-centre</justify>
</vertex_annotate_value>
</string_text>
```

Continue to the next section [28.6.13 Trimesh](#) or return to [28.6 Elements Contained in Models](#) or [28 12d XML File Format](#).

## 28.6.13 Trimesh

A trimesh is a special type of `primitive_3` object.

A trimesh is made up of 3D triangles and can be described by giving the list of  $m$  vertices in the trimesh and the three vertices that make up each of the  $n$  triangular faces.

The 12d XML definition of a trimesh is:

```
<primitive_3d>
  string_header_block
  trimesh_3d_block
</primitive_3d>
```

where

### **string\_header\_block**

the common header block for each string. For the contents and the syntax, see [28.6.3 String Header Block](#).

The **colour** in the `string_header_block` is the base colour for the trimesh.

### **trimesh\_block**

The trimesh block gives the vertices of the trimesh and then the faces of the trimesh in terms of the vertex numbers.

```
<trimesh_3d>
  <vertices>
    <v> x_value_1  y_value_1  z_value_1 </v>
    <v> x_value_2  y_value_2  z_value_2 </v>
    ...
    <v> x_value_n  y_value_n  z_value_n </v>
  </vertices>
  <faces>
    <f> face_1_vertex_1  face_1_vertex_2  face_1_vertex_3 </f>
    <f> face_2_vertex_1  face_2_vertex_2  face_2_vertex_3 </f>
    ...
    <f> face_m_vertex_1  face_m_vertex_2  face_m_vertex_3 </f>
  </vertices>
</trimesh_3d>
```

where

$n$  is the number of vertices and  $(x\_value\_i, y\_value\_i, z\_value\_i)$  are the 3D coordinates of the  $i$ 'th vertex

$m$  is the number of faces in the trimesh and  $face\_j\_vertex\_1, face\_j\_vertex\_2, face\_j\_vertex\_3$  are the number of the vertices in the **vertex** block for the  $j$ 'th face.

Return to [28.6 Elements Contained in Models](#) or [28 12d XML File Format](#)