12d® Model™

ADAC in 12d Model 11

Version 11

May 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   web www.12d.com

ADAC 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

1 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/

See

- 1.1 ADAC.XML Overview
- 1.2 12d Approach to ADAC XML
- 1.4 12d ADAC Menu

1.1 ADAC.XML Overview

See

1.1.1 ADAC XML Structure

The ADAC Schema is fully described by the *ADAC XSD* (**X**ML **S**chema **D**efinition), 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/).

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.

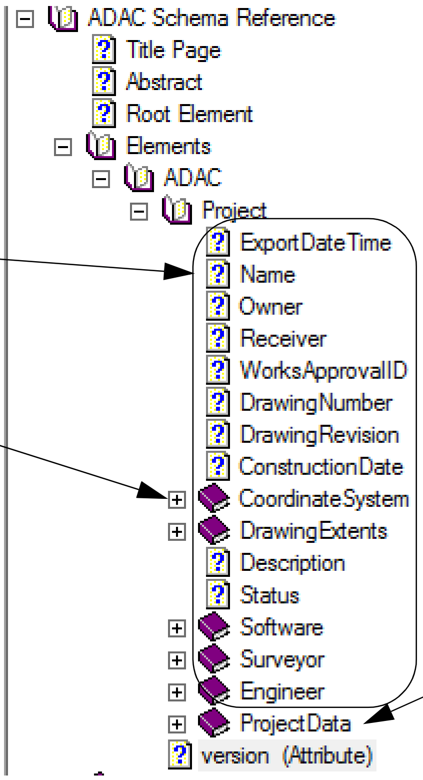
Some of these element are just single values (for example, *ExportDateTime* and *Owner*, and others, contain multi-levelled substructures of data (for example, *Surveyor*, *Engineer and ProjectData*).

Single values are shown with **?** before them

The **+** indicates that the element is not a single value but contains a multi-levelled substructure of values

NOTE:

The image is from the ADAC Version: 4.1.0 Schema Help File



- ADAC Schema Reference
  - Title Page
  - Abstract
  - Root Element
  - Elements
    - ADAC
      - Project
        - ExportDateTime
        - Name
        - Owner
        - Receiver
        - WorksApprovalID
        - DrawingNumber
        - DrawingRevision
        - ConstructionDate
        - CoordinateSystem
        - DrawingExtents
        - Description
        - Status
        - Software
        - Surveyor
        - Engineer
        - ProjectData
        - version (Attribute)

Header Data - common to all the elements under *Project Data*

ProjectData - common to all the elements under Project Data values

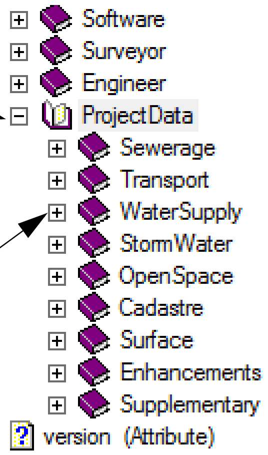
We will refer to all of these elements except **ProjectData** as **Header Data**.

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.*

The **-** indicates that the element contains a multi-levelled substructure and it has been expanded in the diagram



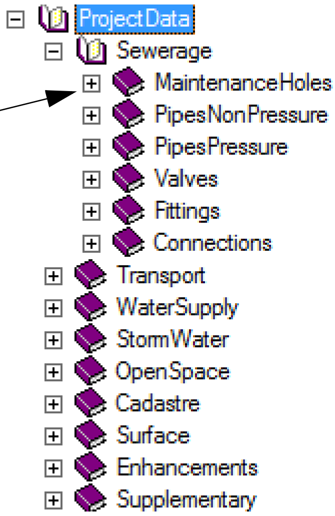
- Software
- Surveyor
- Engineer
- ProjectData
  - Sewerage
  - Transport
  - WaterSupply
  - StormWater
  - OpenSpace
  - Cadastre
  - Surface
  - Enhancements
  - Supplementary
- version (Attribute)

The **+** indicates that the element contains a multi-levelled substructure and it has not been expanded in the diagram

And each of the nine subelements have their own substructures

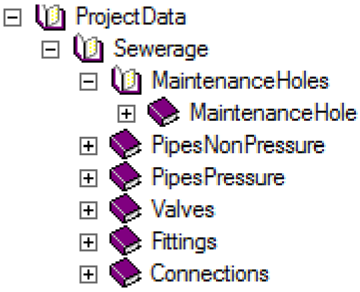
For example, **Sewerage** has the sub elements *MaintenanceHoles, PipesNonPressure, PipesPressure, Valves, Fittings* and *Connections*

The **+** indicates that the element **MaintenanceHoles** contains a multi-levelled substructure



```
⊟ ProjectData
   ⊟ Sewerage
      ⊞ MaintenanceHoles
      ⊞ PipesNonPressure
      ⊞ PipesPressure
      ⊞ Valves
      ⊞ Fittings
      ⊞ Connections
   ⊞ Transport
   ⊞ WaterSupply
   ⊞ StormWater
   ⊞ OpenSpace
   ⊞ Cadastre
   ⊞ Surface
   ⊞ Enhancements
   ⊞ Supplementary
```

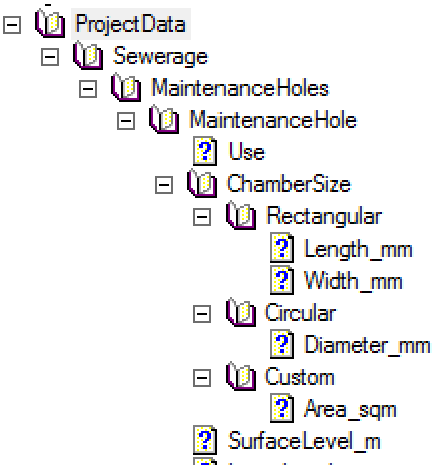
And **MaintenanceHoles** has a substructure that can be expanded out although it has only the one substructure, **MaintenanceHole**.



```
⊟ ProjectData
   ⊟ Sewerage
      ⊟ MaintenanceHoles
         ⊞ MaintenanceHole
      ⊞ PipesNonPressure
      ⊞ PipesPressure
      ⊞ Valves
      ⊞ Fittings
      ⊞ Connections
```

MaintenanceHole itself can be further expanded and part of **MaintenanceHole** is

.



```
⊟ ProjectData
   ⊟ Sewerage
      ⊟ MaintenanceHoles
         ⊟ MaintenanceHole
              ? Use
            ⊟ ChamberSize
               ⊟ Rectangular
                    ? Length_mm
                    ? Width_mm
               ⊟ Circular
                    ? Diameter_mm
               ⊟ Custom
                    ? Area_sqm
              ? SurfaceLevel_m
```

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,

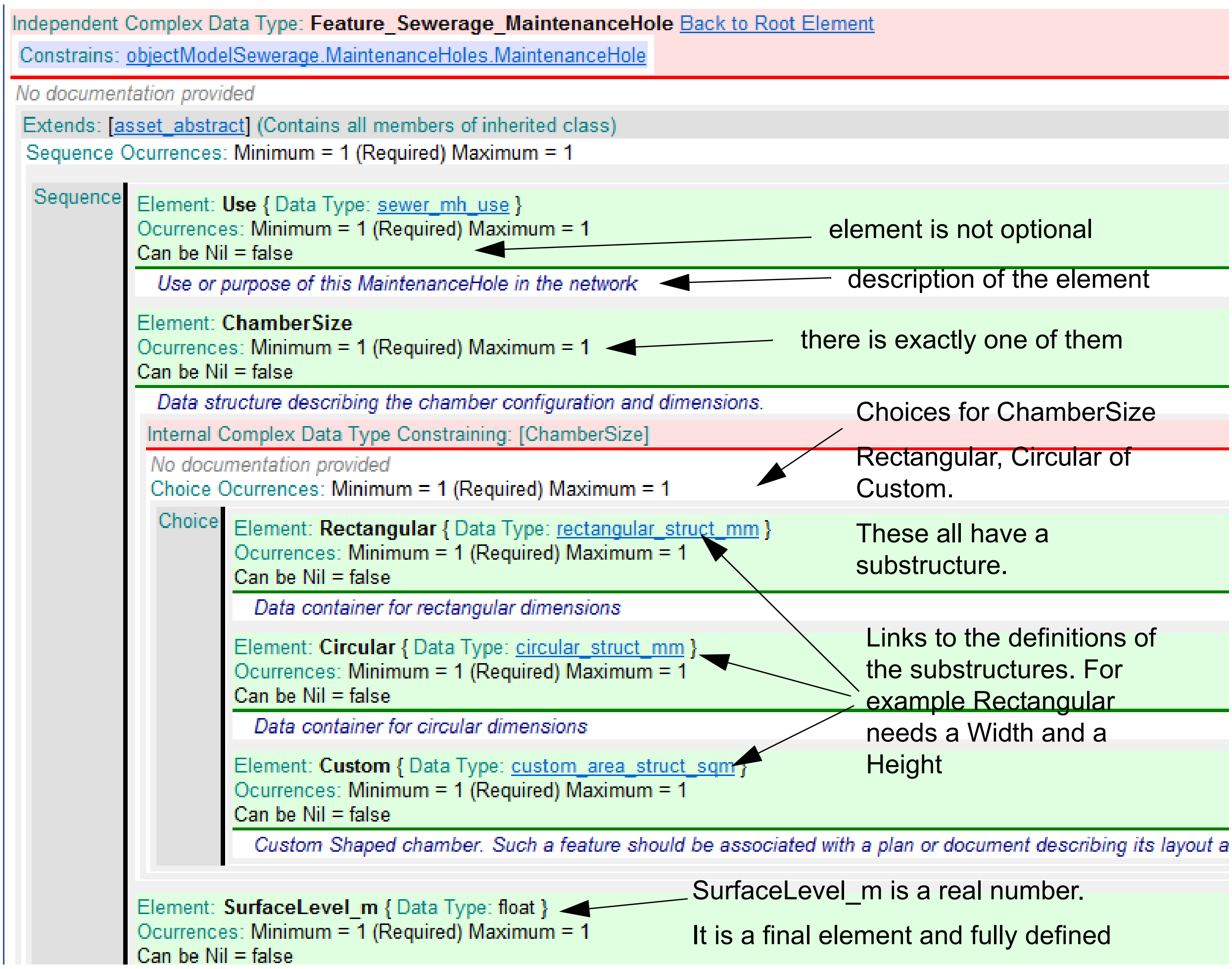
(a) their type - text, integer, real (float) or choices of these.

(b) what ranges the values may, or may not have

(c) if choices, then the list of possible choices (enumerations)

(d) 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 Ocurrences: Minimum = 1 (Required) Maximum = 1

Sequence

Element: **Use** { Data Type: sewer_mh_use }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

Use or purpose of this MaintenanceHole in the network

Element: **ChamberSize**
Ocurrences: 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 Ocurrences: Minimum = 1 (Required) Maximum = 1

Choice

Element: **Rectangular** { Data Type: rectangular_struct_mm }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

Data container for rectangular dimensions

Element: **Circular** { Data Type: circular_struct_mm }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

Data container for circular dimensions

Element: **Custom** { Data Type: custom_area_struct_sqm }
Ocurrences: 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 a

Element: **SurfaceLevel_m** { Data Type: float }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

element is not optional

description of the element

there is exactly one of them

Choices for ChamberSize

Rectangular, Circular of Custom.

These all have a substructure.

Links to the definitions of the substructures. For example Rectangular needs a Width and a Height

SurfaceLevel_m is a real number.

It is a final element and fully defined

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**.

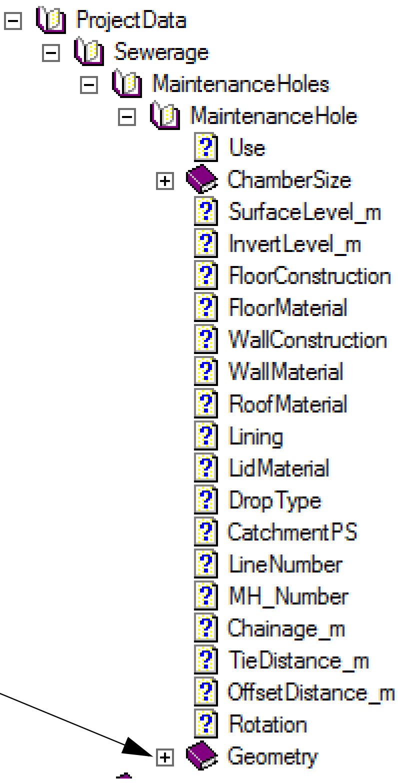
Continue to the next section 1.1.2 ADAC Assets or return to 1.1 ADAC.XML Overview or 1 ADAC .

1.1.2 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**.



- ⊟ ProjectData
  - ⊟ Sewerage
    - ⊟ MaintenanceHoles
      - ⊟ MaintenanceHole
        - Use
        - ⊞ ChamberSize
        - SurfaceLevel_m
        - InvertLevel_m
        - FloorConstruction
        - FloorMaterial
        - WallConstruction
        - WallMaterial
        - RoofMaterial
        - Lining
        - LidMaterial
        - DropType
        - CatchmentPS
        - LineNumber
        - MH_Number
        - Chainage_m
        - TieDistance_m
        - OffsetDistance_m
        - Rotation
        - ⊞ Geometry

The presence of **Geometry** indicates that this element **MaintenanceHole** is a fundamental ADAC element representing an **ADAC Asset**

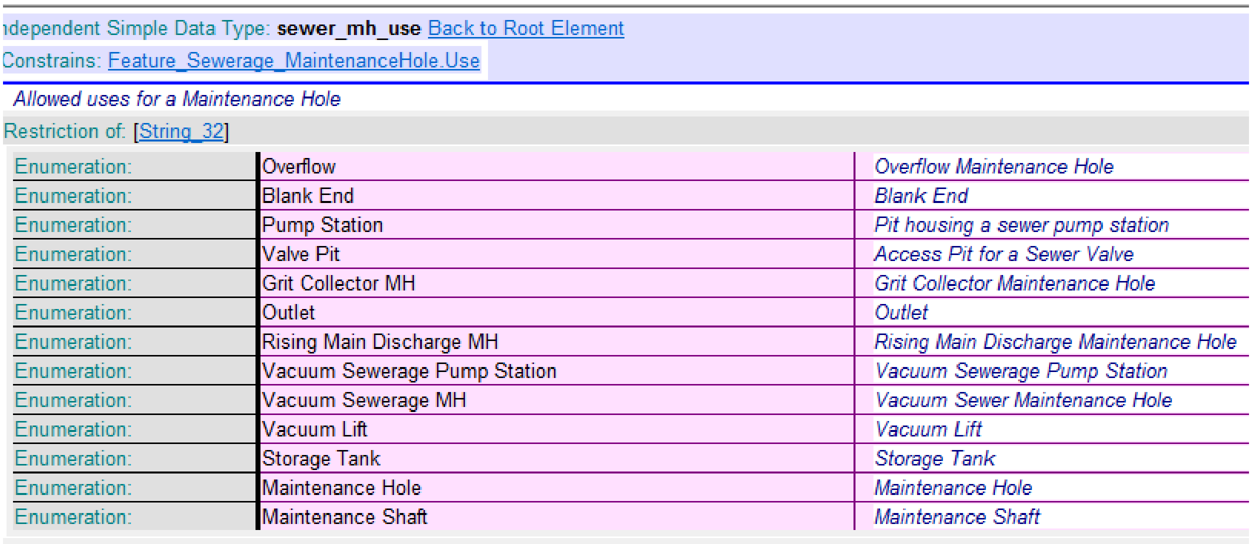
So although **MaintenanceHole** has the substructure **ChamberSize** that can be further expanded, the presence of the **Geometry** element says that **MaintenanceHole** is a fundamental ADAC element representing an **ADAC Asset**.

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):



ndependent 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	*Overflow Maintenance Hole*
Enumeration:	Blank End	*Blank End*
Enumeration:	Pump Station	*Pit housing a sewer pump station*
Enumeration:	Valve Pit	*Access Pit for a Sewer Valve*
Enumeration:	Grit Collector MH	*Grit Collector Maintenance Hole*
Enumeration:	Outlet	*Outlet*
Enumeration:	Rising Main Discharge MH	*Rising Main Discharge Maintenance Hole*
Enumeration:	Vacuum Sewerage Pump Station	*Vacuum Sewerage Pump Station*
Enumeration:	Vacuum Sewerage MH	*Vacuum Sewer Maintenance Hole*
Enumeration:	Vacuum Lift	*Vacuum Lift*
Enumeration:	Storage Tank	*Storage Tank*
Enumeration:	Maintenance Hole	*Maintenance Hole*
Enumeration:	Maintenance Shaft	*Maintenance Shaft*

Continue to the next section 1.1.3 ADAC Geometry Element or return to 1.1 ADAC.XML Overview or 1 ADAC .

1.1.3 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 1.1.4 Guidelines from an Authority Requesting ADAC.XML or return to 1.1 ADAC.XML Overview or 1 ADAC .

1.1.4 Guidelines from an Authority Requesting ADAC.XML

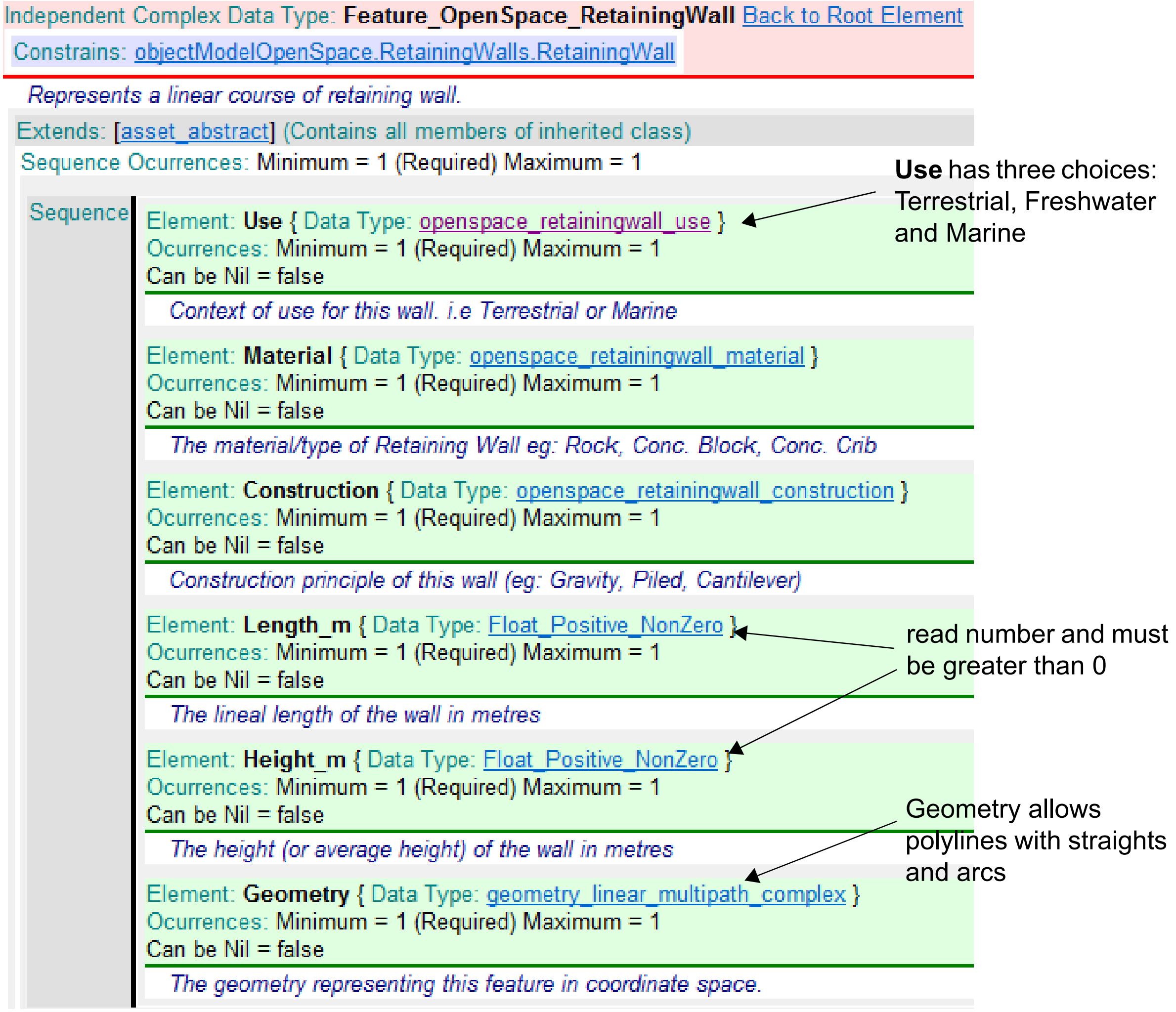
Although ADAC XML has **sixty-eight** Assets, which extends to hundreds of different objects once **Use** and **Type** are considered, not every ADAC Asset is required by every Authority.

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 Ocurrences: Minimum = 1 (Required) Maximum = 1

Sequence

Element: **Use** { Data Type: openspace_retainingwall_use }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

Context of use for this wall. i.e Terrestrial or Marine

Element: **Material** { Data Type: openspace_retainingwall_material }
Ocurrences: 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 }
Ocurrences: 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 }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

The lineal length of the wall in metres

Element: **Height_m** { Data Type: Float_Positive_NonZero }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

The height (or average height) of the wall in metres

Element: **Geometry** { Data Type: geometry_linear_multipath_complex }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

The geometry representing this feature in coordinate space.

Use has three choices: Terrestrial, Freshwater and Marine

read number and must be greater than 0

Geometry allows polylines with straights and arcs

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

From the ADAC.XML Guidelines from Bundaberg, Gladstone and Rockhampton Regional Councils

StormWater

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.

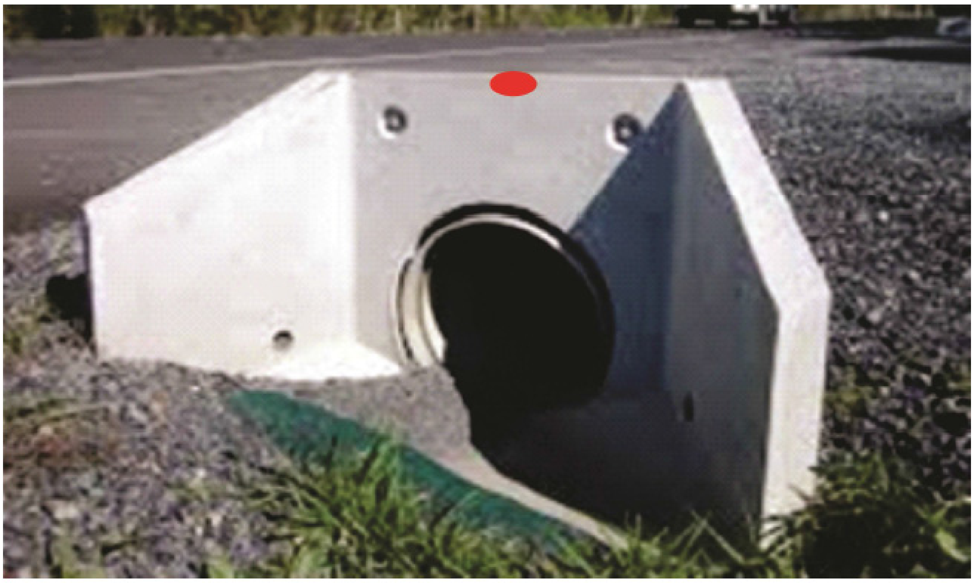


Figure 5

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.

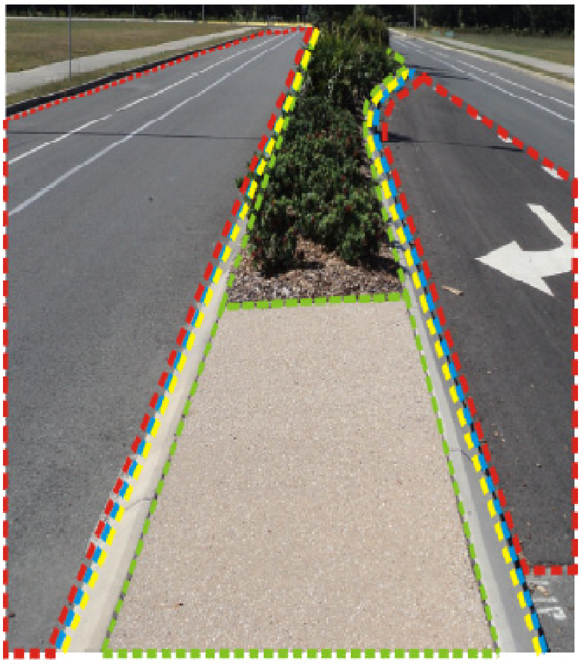


Figure 1

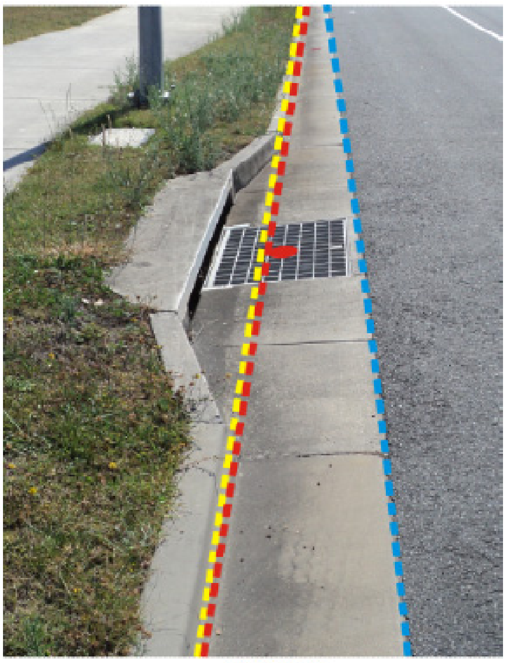


Figure 2

Return to 1.1 ADAC.XML Overview or 1 ADAC .

1.2 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 1.3 12d ADAC Workflow or return to 1 ADAC .

1.3 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 1.6 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.

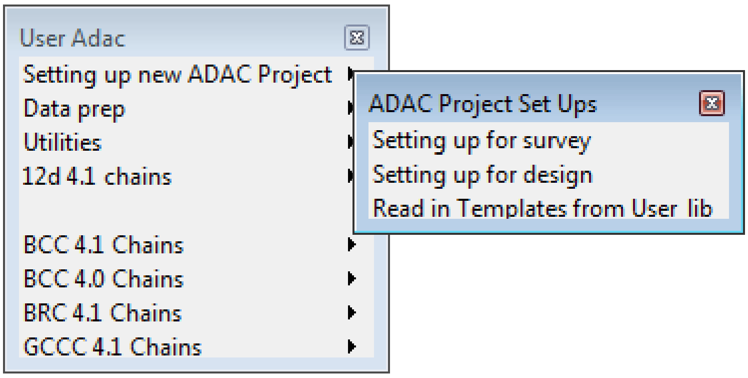
This is done **once for a new project** and only involves clicking a menu item:

File I/O =>ADAC =>User =>Setting up for survey - for surveys

For more information on what this option does, see 1.4.9.1.1 Setting Up for Survey .

File I/O =>ADAC =>User =>Setting up for design - for designers.

For more information on what this option does, see 1.4.9.1.2 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 1.4.1 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 1.4.9.2 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 1.4.2 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.

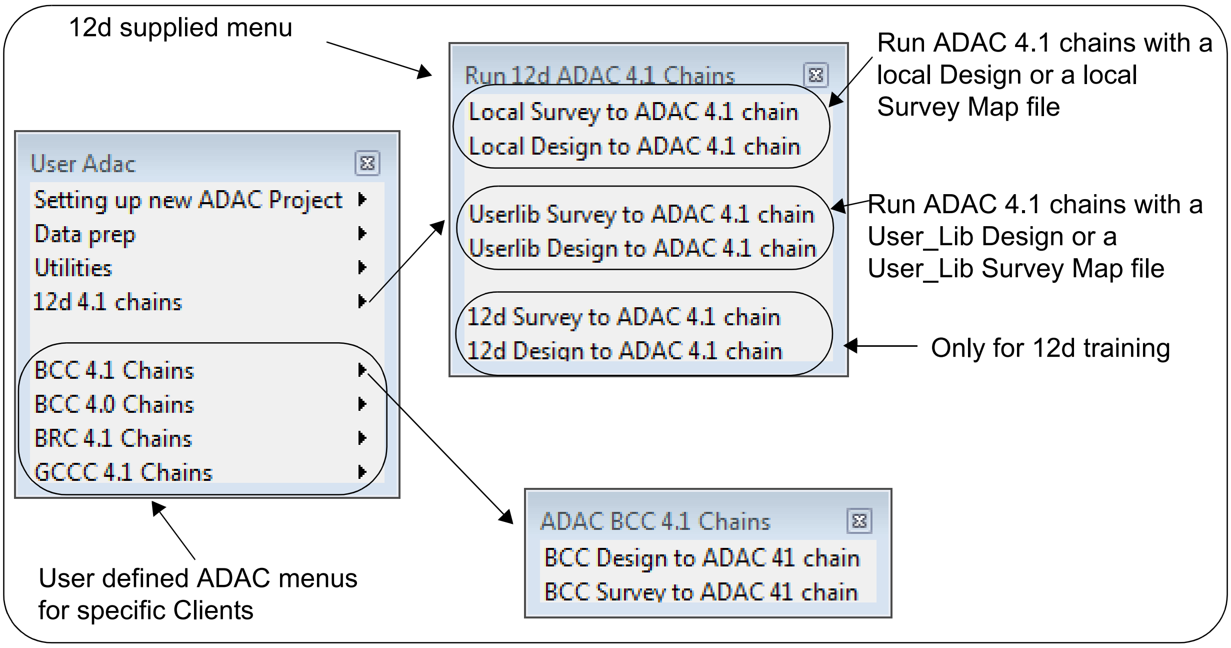
Step 5.Running a Design or Survey Chain - run regularly in an ADAC project

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 1.5 ADAC Design and Survey Chains .


12d supplied menu
Run 12d ADAC 4.1 Chains
Local Survey to ADAC 4.1 chain
Local Design to ADAC 4.1 chain
Userlib Survey to ADAC 4.1 chain
Userlib Design to ADAC 4.1 chain
12d Survey to ADAC 4.1 chain
12d Design to ADAC 4.1 chain
Run ADAC 4.1 chains with a local Design or a local Survey Map file
Run ADAC 4.1 chains with a User_Lib Design or a User_Lib Survey Map file
Only for 12d training
User Adac
Setting up new ADAC Project
Data prep
Utilities
12d 4.1 chains
BCC 4.1 Chains
BCC 4.0 Chains
BRC 4.1 Chains
GCCC 4.1 Chains
ADAC BCC 4.1 Chains
BCC Design to ADAC 41 chain
BCC Survey to ADAC 41 chain
User defined ADAC menus for specific Clients

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 1.4.4 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 1.4.5 Report

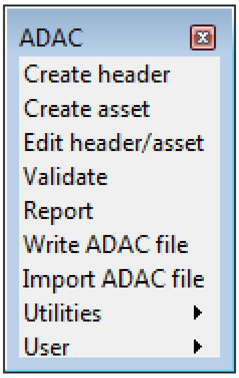
Step 7.Writing the Data out to ADAC XML - done as required

This is a simple process and is done by running just the one panel. See 1.4.6 Write ADAC XML File

Continue to the next section 1.4 12d ADAC Menu or return to 1 ADAC .

1.4 12d ADAC Menu

Position of menu: **File I/O =>ADAC**



See

- 1.4.1 Create Header
- 1.4.2 Create ADAC Asset
- 1.4.3 Edit ADAC Header/Asset
- 1.4.4 Validate
- 1.4.5 Report
- 1.4.6 Write ADAC XML File
- 1.4.7 Import ADAC XML File
- 1.4.8 ADAC Utilities
- 1.4.9 User ADAC

1.4.1 Create Header

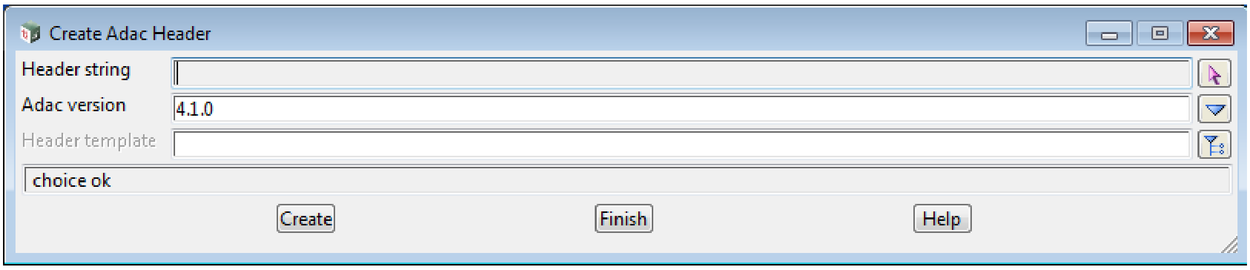
Position of option on menu: **File I/O =>ADAC =>Create header**

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:



Create Adac Header

Header string:

Adac version: 4.1.0

Header template:

choice ok

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

Field Description Type Defaults Pop-Up

Header string string select

select a one vertex string to set up with the ADAC Header Data attributes.

ADAC version choice box from env.4d 4.0.0, 4.1.0

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 **Edit Environment Variables** *panel at* **External Apps >ADAC**. *See* 7.6.3 env.4d .

Header template ADAC Header box available ADAC Header templates

the name of an ADAC Header template that is used to fill in some of the ADAC Header information for the Header string.

Create button

clicking **Create** *creates an ADAC Header structure as 12d ADAC attributes for the string, and if there is a* **Header template**, *it loads the string's ADAC attributes with values from the Header Template.*

The 1.4.2 Create ADAC Asset *panel is then brought up so that the ADAC Header data can be create/edited.*

Continue to the next section 1.4.2 Create ADAC Asset or return to 1.4 12d ADAC Menu or 1 ADAC .

1.4.2 Create ADAC Asset

Position of option on menu: **File I/O =>ADAC =>Create Asset**

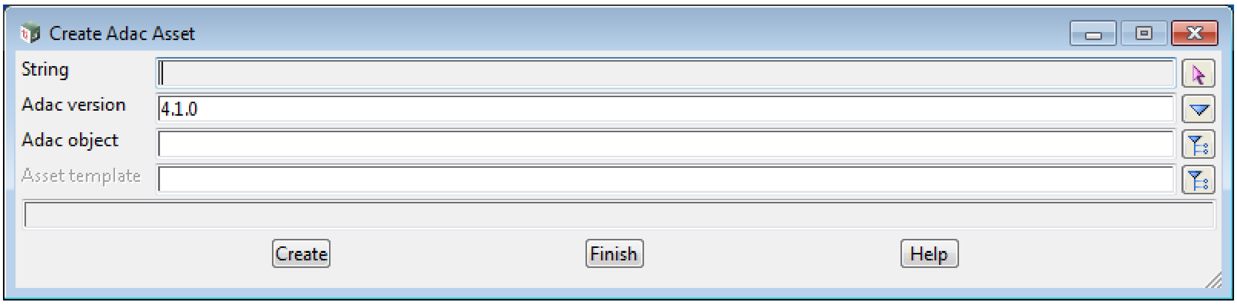
To produce an ADAC XML file, ***12d Model*** needs one string with the ADAC Header data, and zero or more strings with ADAC Asset data.

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
String	string select		

select a string that is to be set up with the ADAC Asset.

ADAC version	choice box	from env.4d	4.0.0, 4.1.0

the version of ADAC to use when creating the ADAC Header data.

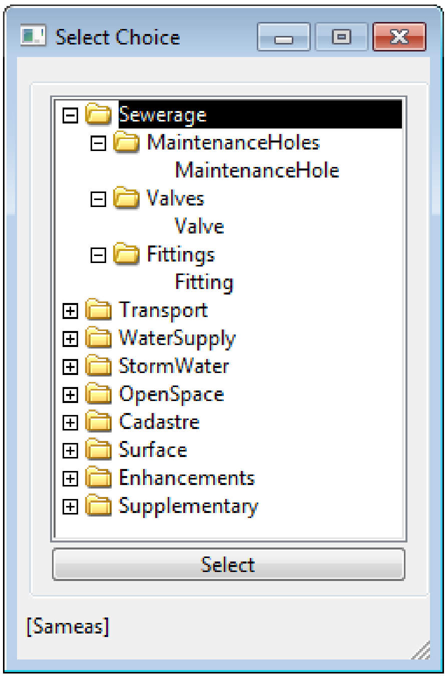
The default value is set by the environment variable ADAC_VERSION_4D which is set in the env.4d editor at **External Apps >ADAC**.

ADAC Asset	choice box	depends on the string geometry

If the geometry of the string selected is a:

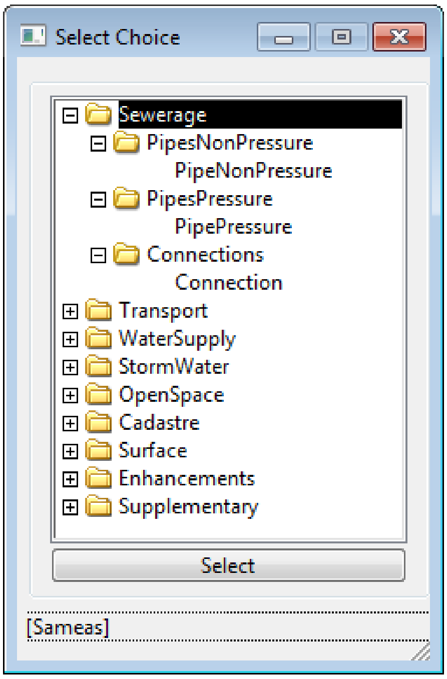
Point *(a one vertex string), then the pop-up only shows those ADAC Assets that have a Point Geometry.*

For example, for ADAC 4.1, the only ADAC Assets in Sewerage with a Point Geometry are Sewerage>MaintenanceHoles>MaintenanceHole, Sewerage>Valves>Valve and Sewerage>Fittings>Fitting



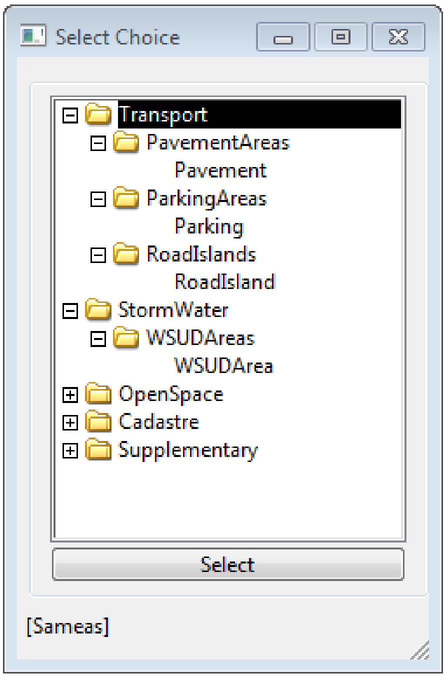
***Polyline** then the pop-up only shows those ADAC Assets that have Polyline Geometry.*

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 1.4.3 Edit ADAC Header/Asset *panel is then brought up so that the ADAC Header data can be create/edited.*

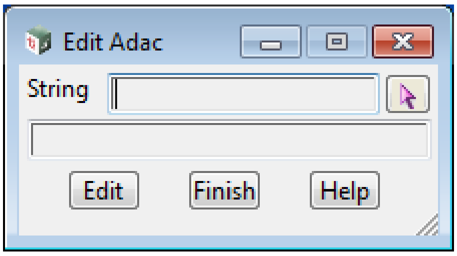
Continue to the next section 1.4.3 Edit ADAC Header/Asset or return to 1.4 12d ADAC Menu or 1 ADAC .

1.4.3 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
String	string select		

select a string to have its ADAC attributes edited.

Edit	button		

if the selected string is a ADAC Header string, then the 1.4.2 Create ADAC Asset *is brought up.*

If the selected string is an ADAC Asset string, the 1.4.3 Edit ADAC Header/Asset *panel is brought up for its ADAC Asset type.*

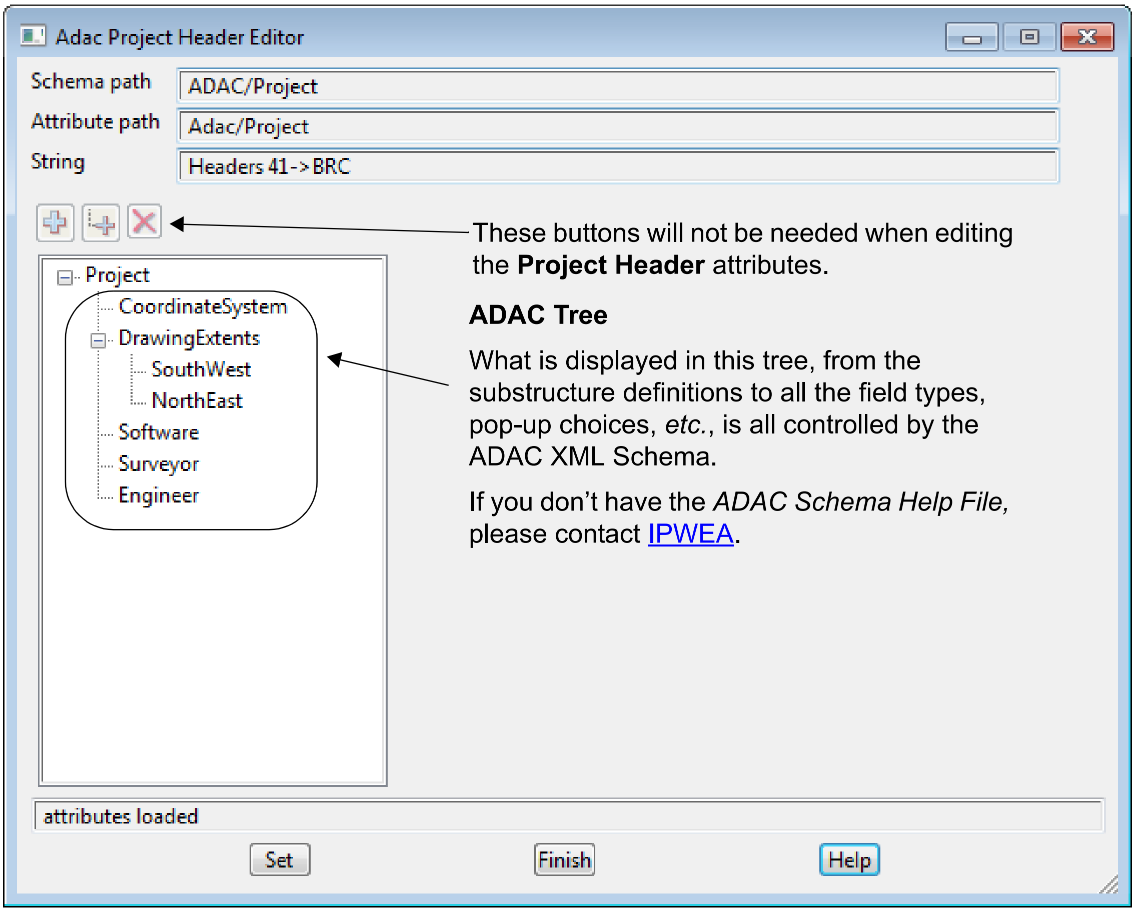
If the selected string is not an ADAC Header or an ADAC Asset string, then an error message is written to the panel message area.

Continue to the next section 1.4.3.1 ADAC Header Editor or return to 1.4.3 Edit ADAC Header/Asset , 1.4 12d ADAC Menu or 1 ADAC .

1.4.3.1 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.



Adac Project Header Editor

Schema path: ADAC/Project

Attribute path: Adac/Project

String: Headers 41->BRC

These buttons will not be needed when editing the **Project Header** attributes.

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.

attributes loaded

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

Field Description	Type	Defaults	Pop-Up
Schema path	output only		

the path to the ADAC Schema that is used to generate all the items under ***Project****.*

Attribute path output only

the path in the ADAC Schema to the ADAC element being edited. In this case it is the ADAC Project element.

String output only

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.

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.*

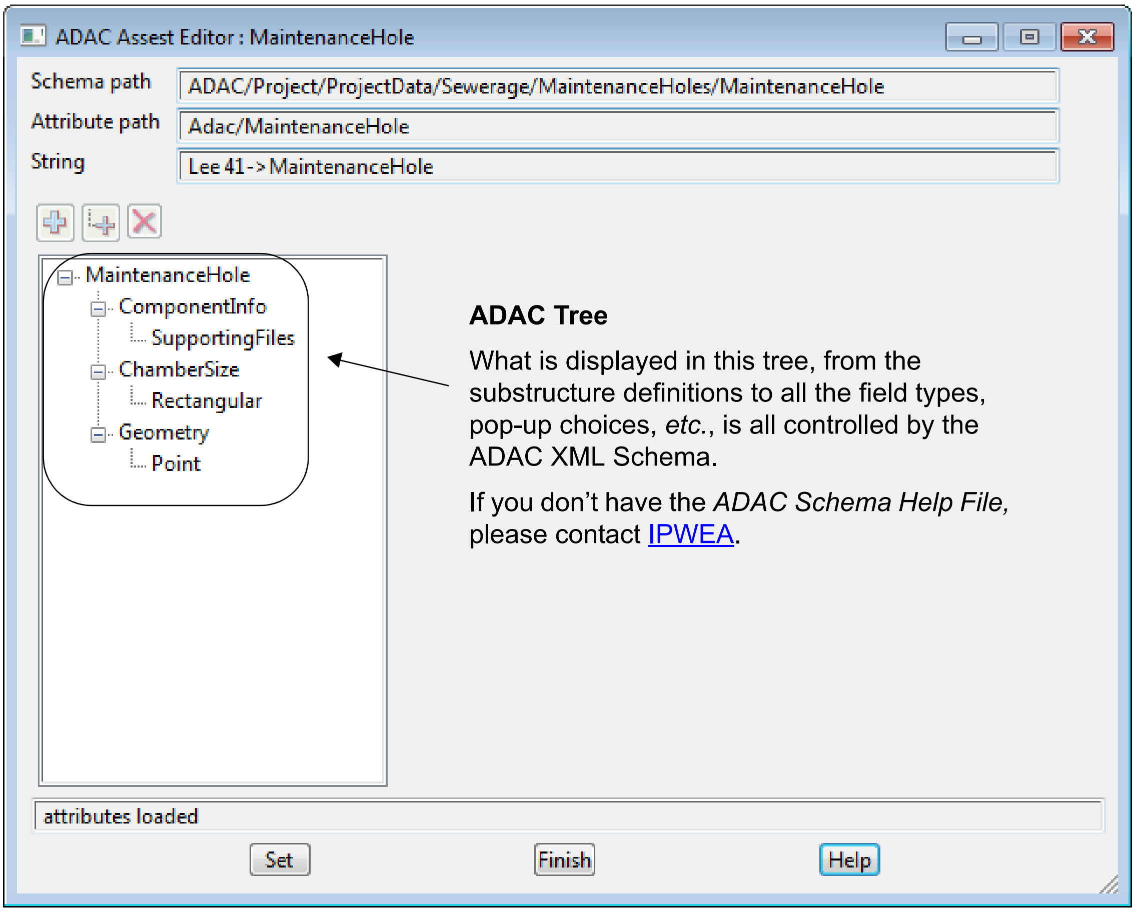
Continue to the next section 1.4.3.2 ADAC Asset Editor or return to 1.4.3 Edit ADAC Header/ Asset , 1.4 12d ADAC Menu or 1 ADAC .

1.4.3.2 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



ADAC Assest Editor : MaintenanceHole

Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole

Attribute path: Adac/MaintenanceHole

String: Lee 41->MaintenanceHole

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.

attributes loaded

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

Field Description	Type	Defaults	Pop-Up
Schema path	output only		

the path to the ADAC Schema that is used to generate all the items for this ADAC Asset.

Attribute path output only

the path in the ADAC Schema to the ADAC element being edited. In this case it is the ADAC Sewerage MaintenanceHole element.

String output only

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.

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 1.4.3.3 Common Features of ADAC Editors or return to 1.4.3 Edit ADAC Header/Asset , 1.4 12d ADAC Menu or 1 ADAC .

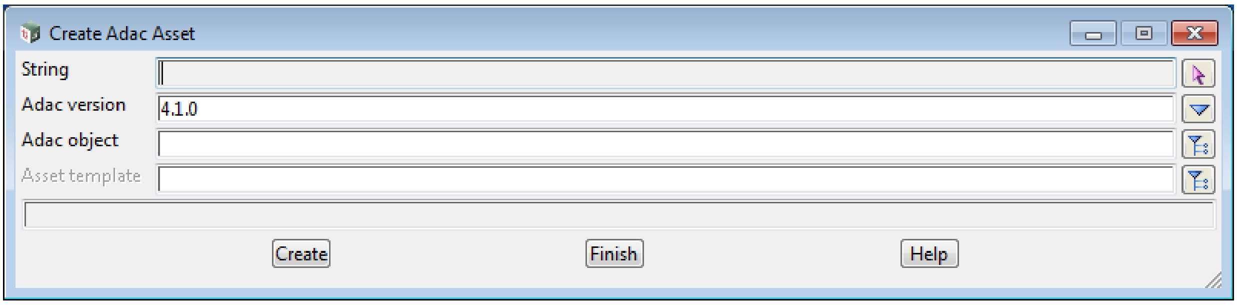
1.4.3.3 Common Features of ADAC Editors

The *ADAC Schema* is fully described by the *ADAC XSD* (**X**ML **S**chema **D**efinition), 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:



Create Adac Asset

String

Adac version 4.1.0

Adac object

Asset template

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

1.4.3.3.1 Editor Tree, Nodes and Elements

1.4.3.3.2 Delete Icon

1.4.3.3.3 Add Sibling and Add Child Icons

1.4.3.3.4 Nillable Elements and Nodes

1.4.3.3.5 Nodes with Choices for Subnodes

1.4.3.3.1 Editor Tree, Nodes and Elements

When the ADAC **Editors** start up, they have a tree structure on the left hand side.

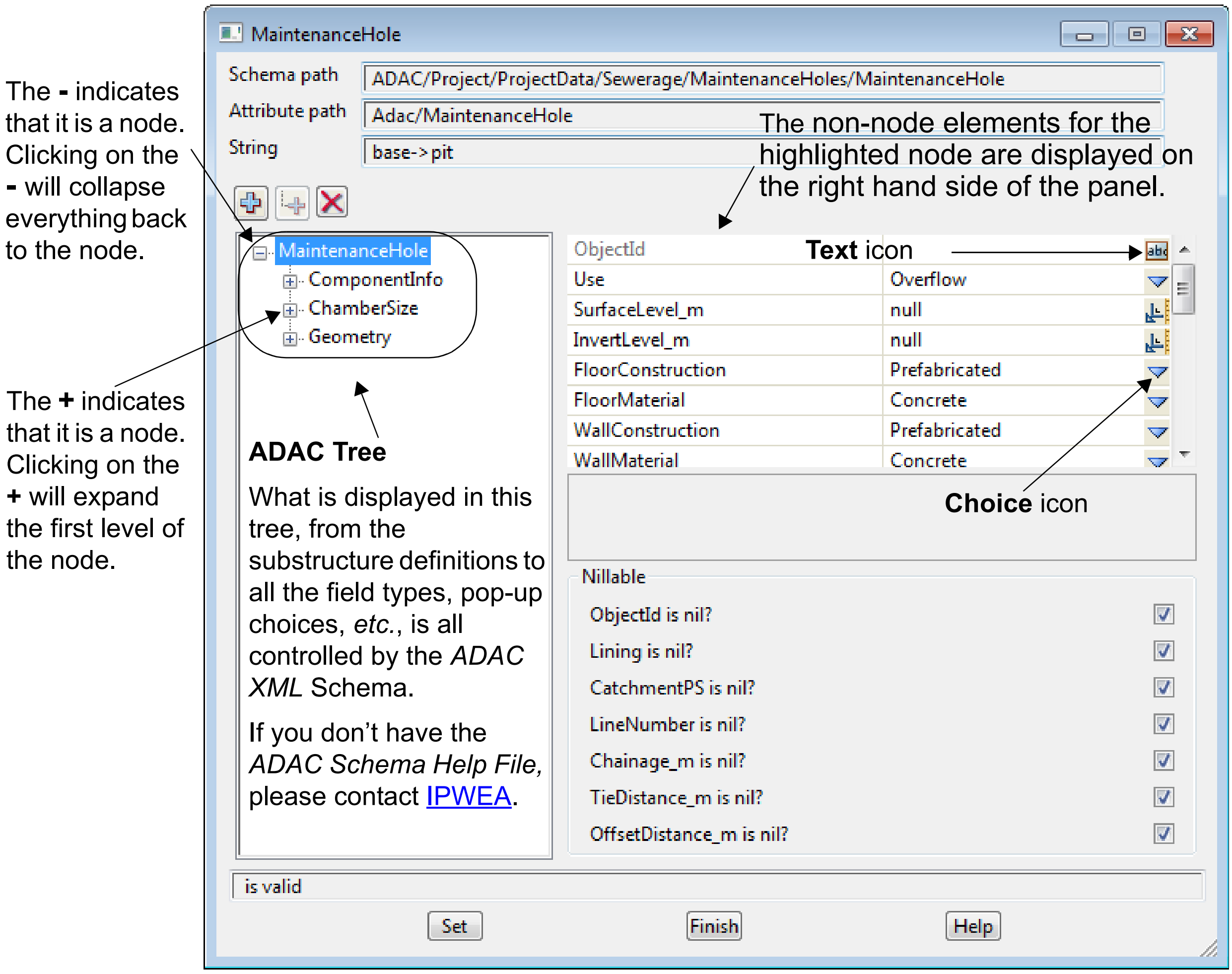
Each item in the tree are known as nodes and **nodes** contain ADAC elements or can have their own substructure of nodes and elements. A node with a substructure is identified by having either a **+** or a **-** to the left of it.

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.

Text icon

Choice icon

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.

MaintenanceHole

Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole

Attribute path: Adac/MaintenanceHole

String: base->pit

- MaintenanceHole
  - ComponentInfo
  - ChamberSize
  - Geometry

ObjectId	
Use	Overflow
SurfaceLevel_m	null
InvertLevel_m	null
FloorConstruction	Prefabricated
FloorMaterial	Concrete
WallConstruction	Prefabricated
WallMaterial	Concrete

Nillable

ObjectId is nil? ☑

Lining is nil? ☑

CatchmentPS is nil? ☑

LineNumber is nil? ☑

Chainage_m is nil? ☑

TieDistance_m is nil? ☑

OffsetDistance_m is nil? ☑

is valid

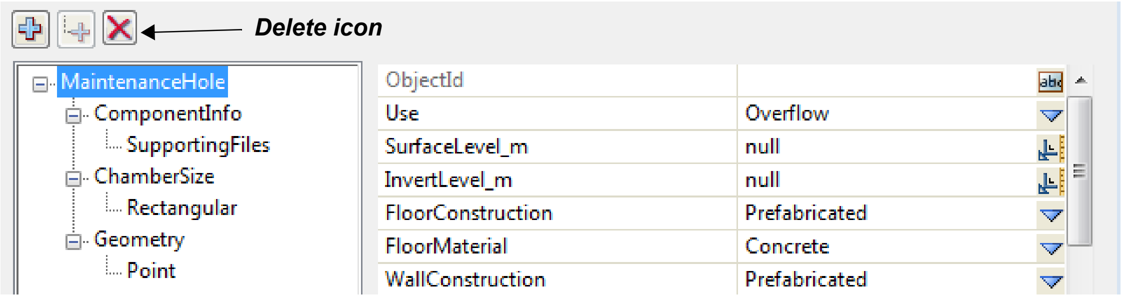
The elements on the right hand side of the panel mainly consist of choice, text, real, integer and date boxes and they are all defined in the ***ADAC XSD***. Where ever possible, the icons at the end of each field are the same as those used in ***12d Model*** panels.

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 1.4.3.3.2 Delete Icon or return to 1.4.3.3 Common Features of ADAC Editors , 1.4.3 Edit ADAC Header/Asset , 1.4 12d ADAC Menu or 1 ADAC .

1.4.3.3.2 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.



Delete icon

- MaintenanceHole
  - ComponentInfo
    - SupportingFiles
  - ChamberSize
    - Rectangular
  - Geometry
    - Point

ObjectId	
Use	Overflow
SurfaceLevel_m	null
InvertLevel_m	null
FloorConstruction	Prefabricated
FloorMaterial	Concrete
WallConstruction	Prefabricated

If a node in the tree is selected and highlighted, and the **Delete** icon is enabled and then clicked, the node is deleted.

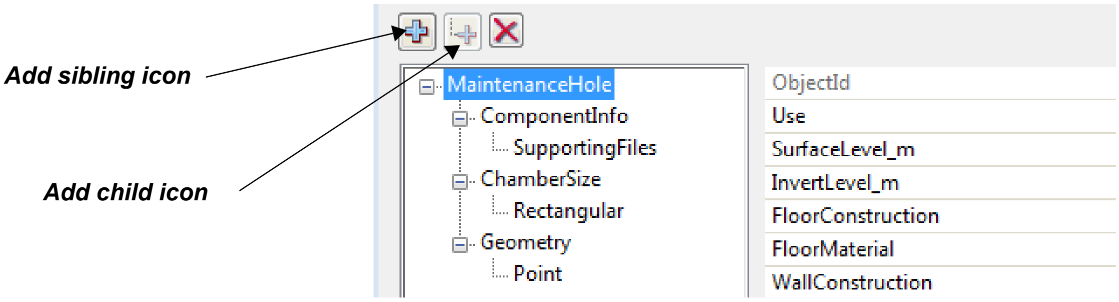
The **Delete** icon is enabled when the *ADAC Asset* itself has been selected as in the image above but deleting the *ADAC Asset* itself is rarely done.

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 1.4.3.3.5 Nodes with Choices for Subnodes .

Continue to the next section 1.4.3.3.3 Add Sibling and Add Child Icons or return to 1.4.3.3 Common Features of ADAC Editors , 1.4.3 Edit ADAC Header/Asset , 1.4 12d ADAC Menu or 1 ADAC .

1.4.3.3.3 Add Sibling and Add Child Icons

There is 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 1.4.3.3.5 Nodes with Choices for Subnodes .

Continue to the next section 1.4.3.3.4 Nillable Elements and Nodes or return to 1.4.3.3 Common Features of ADAC Editors , 1.4.3 Edit ADAC Header/Asset , 1.4 12d ADAC Menu or 1 ADAC .

1.4.3.3.4 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

1.4.3.3.4.1 Nillable Elements

1.4.3.3.4.2 Nillable Node

Or return to 1.4.3.3 Common Features of ADAC Editors , 1.4.3 Edit ADAC Header/Asset , 1.4 12d ADAC Menu or 1 ADAC .

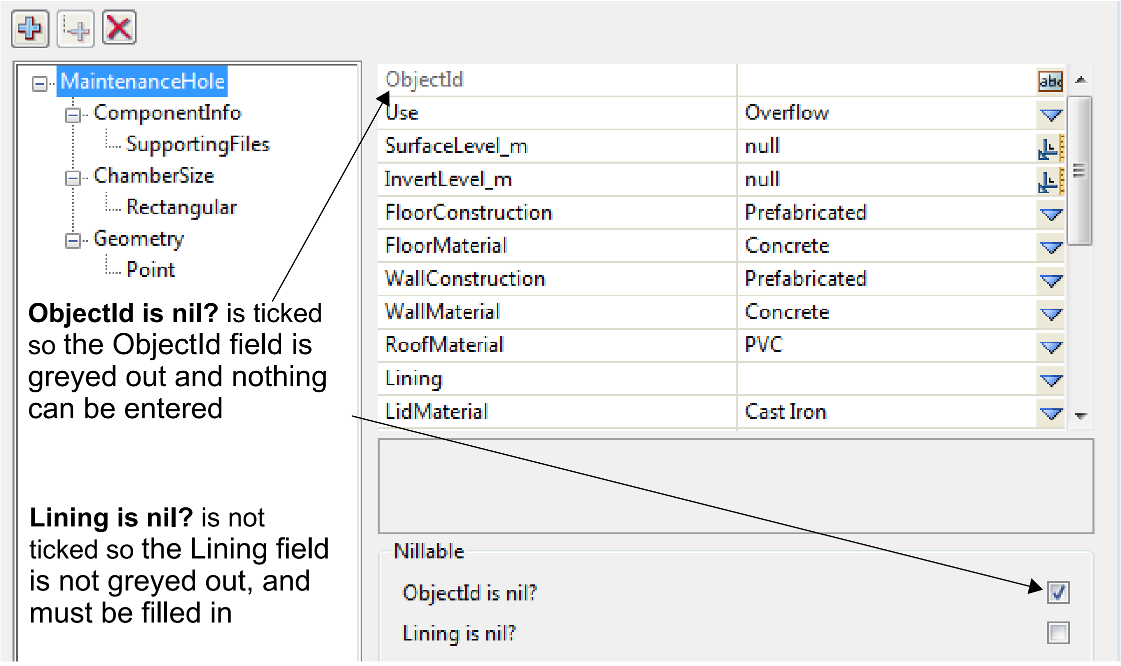
1.4.3.3.4.1 Nillable Elements

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. In this is the case a value must be entered.

All elements not mentioned in the Nillable area are **not nillable** and must always be filled in.



ObjectId	
Use	Overflow
SurfaceLevel_m	null
InvertLevel_m	null
FloorConstruction	Prefabricated
FloorMaterial	Concrete
WallConstruction	Prefabricated
WallMaterial	Concrete
RoofMaterial	PVC
Lining	
LidMaterial	Cast Iron

Nillable

ObjectId is nil? ☑

Lining is nil? □

ObjectId is nil? is ticked so the ObjectId field is greyed out and nothing can be entered

Lining is nil? is not ticked so the Lining field is not greyed out, and must be filled in

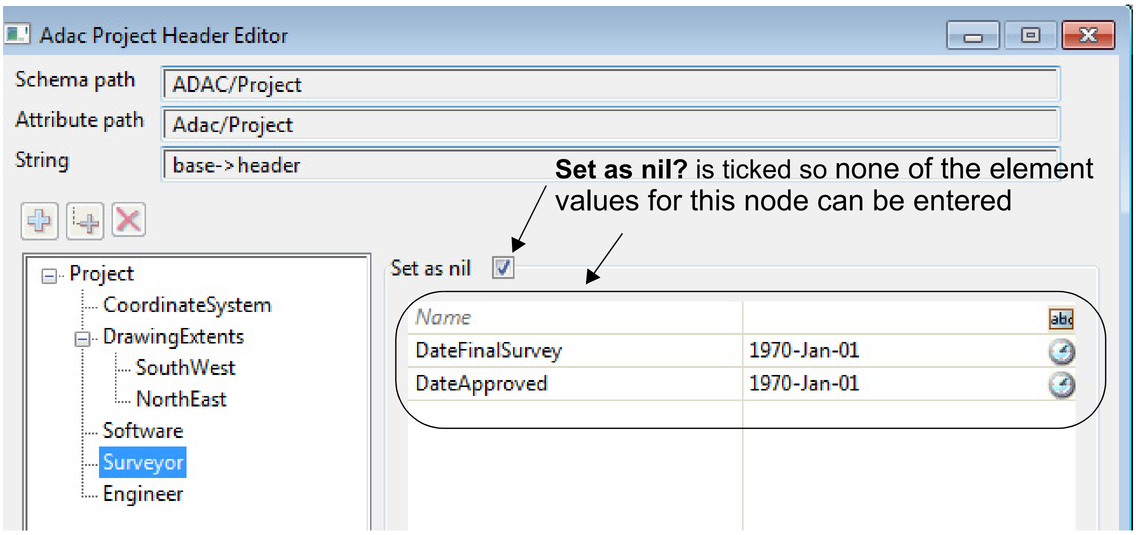
Continue to the next section 1.4.3.3.4.2 Nillable Node or return to 1.4.3.3.4 Nillable Elements and Nodes .

1.4.3.3.4.2 Nillable Node

In the ADAC **Editor**, if an 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.



Schema path: ADAC/Project
Attribute path: Adac/Project
String: base->header

Set as nil ☑

Name	
DateFinalSurvey	1970-Jan-01
DateApproved	1970-Jan-01

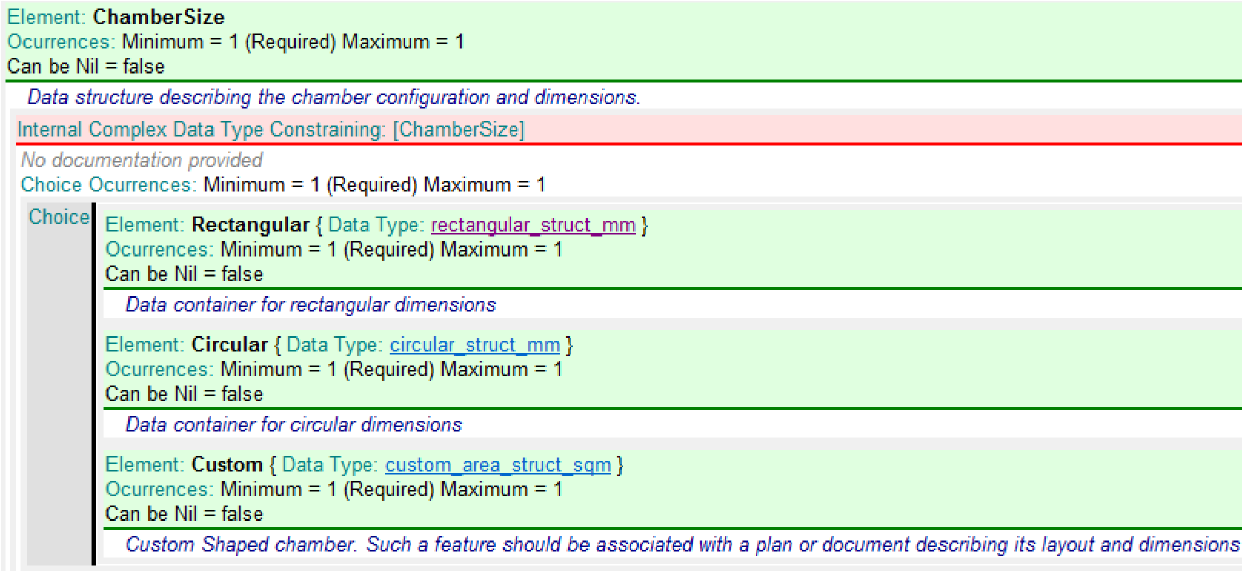
Set as nil? is ticked so none of the element values for this node can be entered

Return to 1.4.3.3.4 Nillable Elements and Nodes .

1.4.3.3.5 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**
Ocurrences: 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 Ocurrences: Minimum = 1 (Required) Maximum = 1

Choice

Element: **Rectangular** { Data Type: rectangular_struct_mm }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

Data container for rectangular dimensions

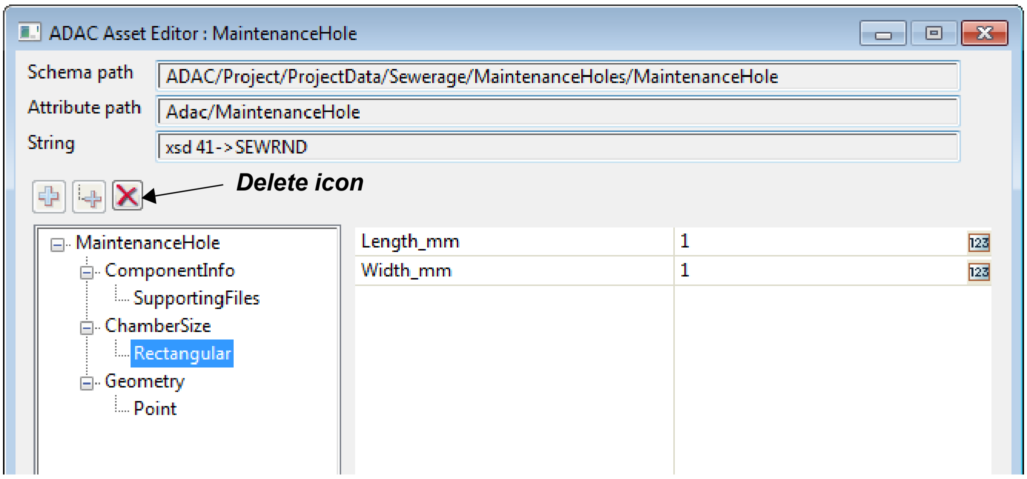
Element: **Circular** { Data Type: circular_struct_mm }
Ocurrences: Minimum = 1 (Required) Maximum = 1
Can be Nil = false

Data container for circular dimensions

Element: **Custom** { Data Type: custom_area_struct_sqm }
Ocurrences: 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

-



ADAC Asset Editor : MaintenanceHole

Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole
Attribute path: Adac/MaintenanceHole
String: xsd 41->SEWRND

Delete icon

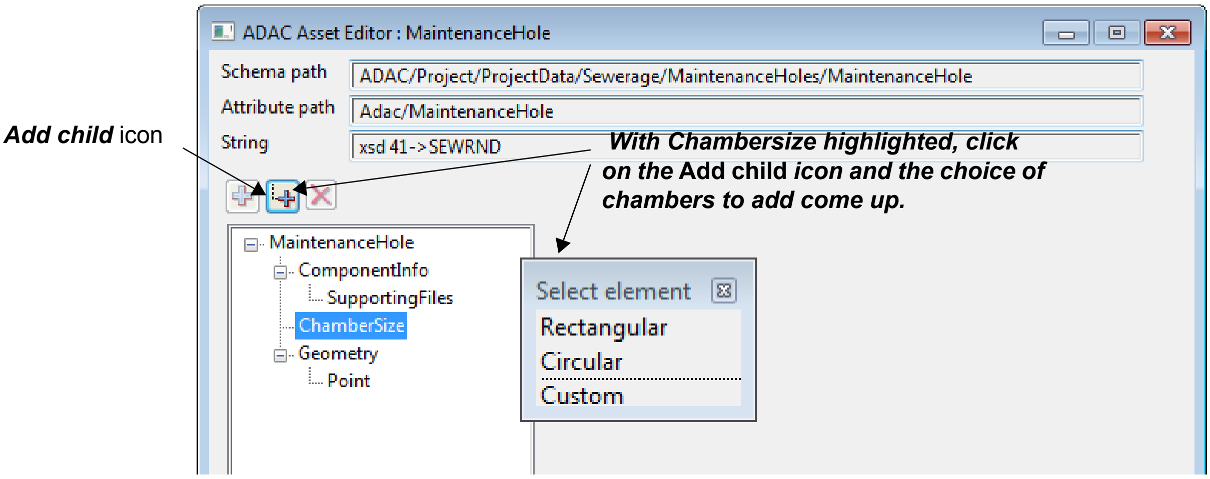
Length_mm	1
Width_mm	1

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*.

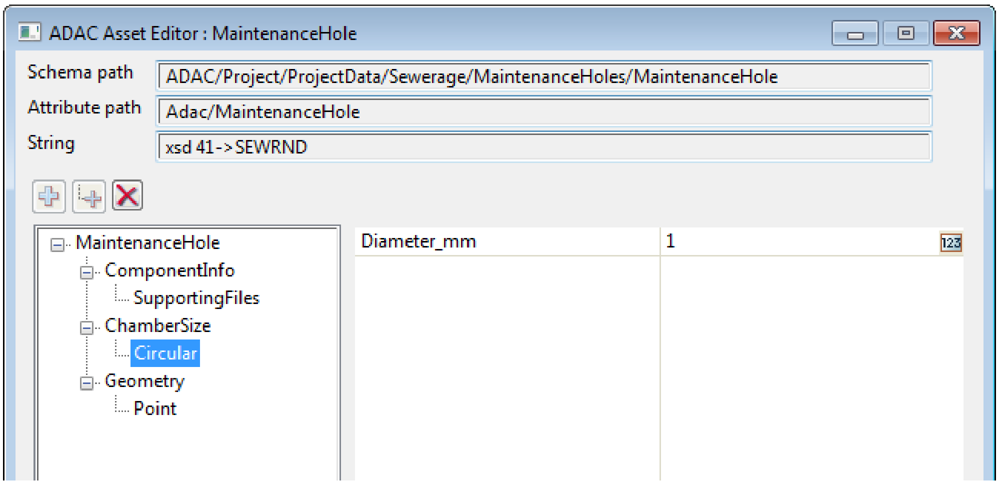
-



With Chambersize highlighted, click on the* Add child *icon and the choice of chambers to add come up.

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*



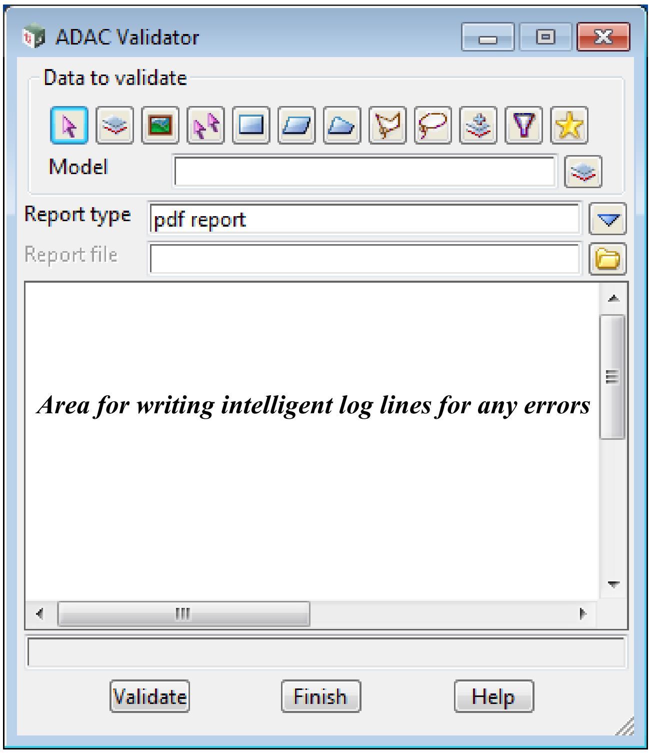
Return to 1.4.3.3 Common Features of ADAC Editors .

1.4.4 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:



Report type pdf report

Area for writing intelligent log lines for any errors

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 validated against the ADAC Schema.

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 but errors are still written to the panel's message area.*

Validate button

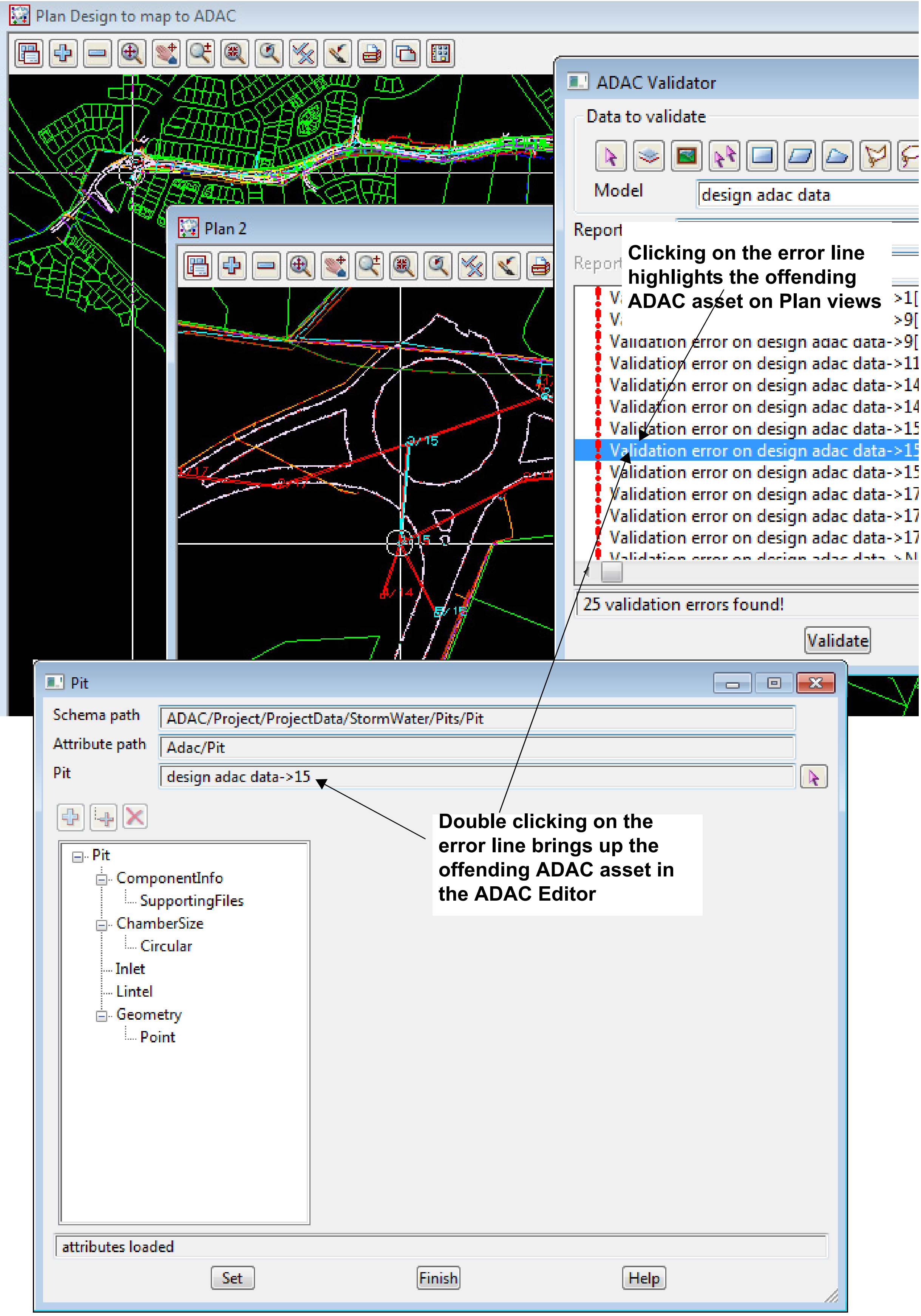
when **Validate** *is pressed, the string in the Data Source is validated against the ADAC XML Schema.*

*If no errors are found, then **No errors found!** is written to the panel's message area.*

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 **Report file** *is not blank, written in XML to the report file.*

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.



Model design adac data

Validation error on design adac data->9[
Validation error on design adac data->11
Validation error on design adac data->14
Validation error on design adac data->14
Validation error on design adac data->15
Validation error on design adac data->15
Validation error on design adac data->15
Validation error on design adac data->17
Validation error on design adac data->17
Validation error on design adac data->17

25 validation errors found!

Schema path ADAC/Project/ProjectData/StormWater/Pits/Pit

Attribute path Adac/Pit

Pit design adac data->15

```
Pit
  ComponentInfo
    SupportingFiles
  ChamberSize
    Circular
  Inlet
  Lintel
  Geometry
    Point
```

attributes loaded

Clicking on the error line highlights the offending ADAC asset on Plan views

Double clicking on the error line brings up the offending ADAC asset in the ADAC Editor

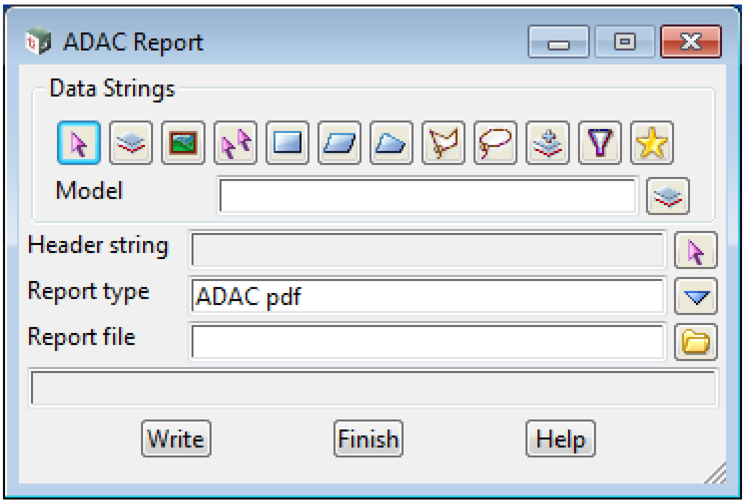
Continue to the next section 1.4.5 Report or return to 1.4 12d ADAC Menu .

1.4.5 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:



Report type: ADAC pdf

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.

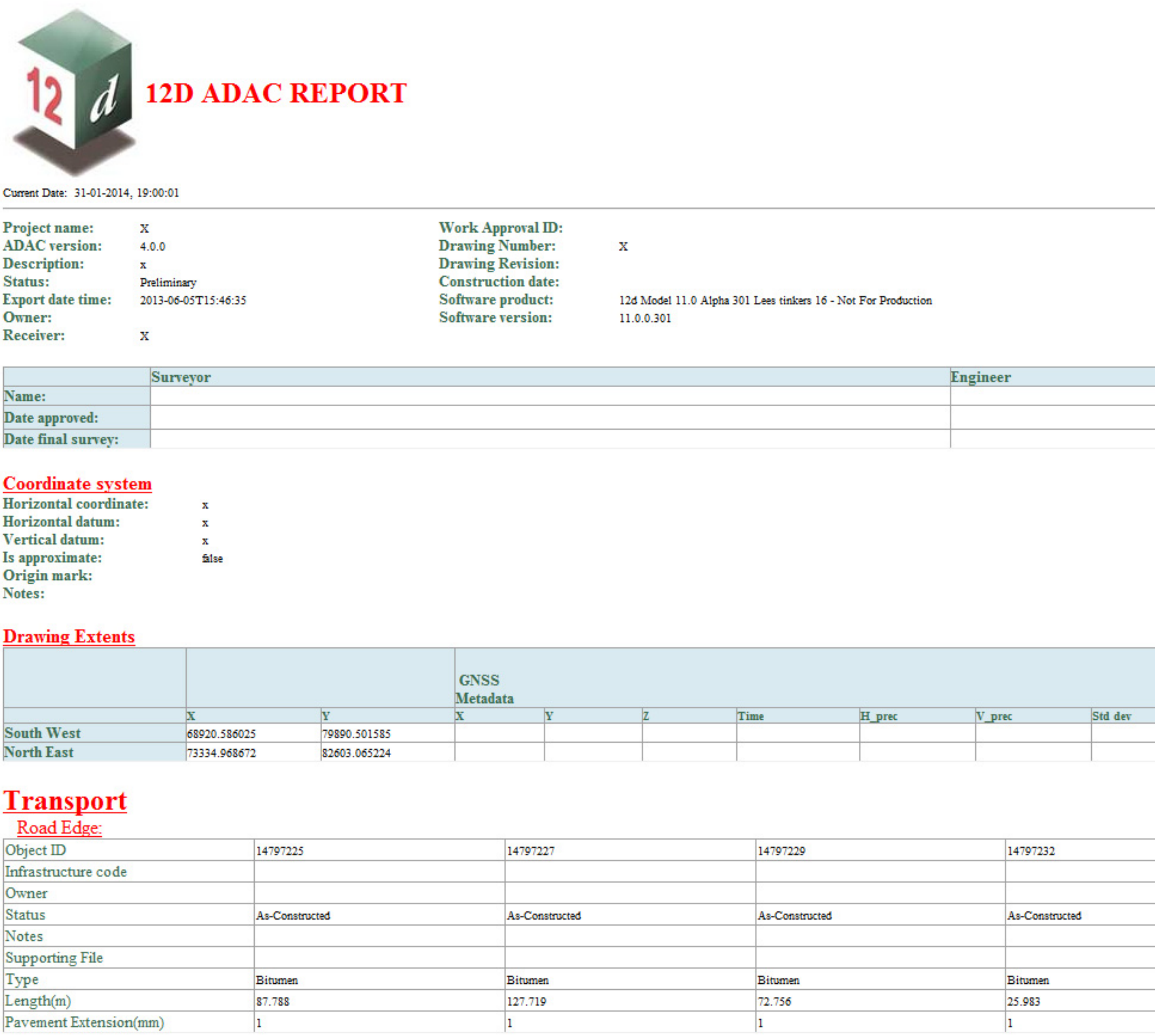
For information on setting up custom reports from the generated XML file using xslts, see 4.30 Setting Up XML Reports .

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

Project name:	x	**Work Approval ID:**	
ADAC version:	4.0.0	**Drawing Number:**	x
Description:	x	**Drawing Revision:**	
Status:	Preliminary	**Construction date:**	
Export date time:	2013-06-05T15:46:35	**Software product:**	12d Model 11.0 Alpha 301 Lees tinkers 16 - Not For Production
Owner:		**Software version:**	11.0.0.301
Receiver:	x		

	Surveyor	Engineer
Name:		
Date approved:		
Date final survey:		

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
South West	68920.586025	79890.501585							
North East	73334.968672	82603.065224							

Transport

Road Edge:

Object ID	14797225	14797227	14797229	14797232
Infrastructure code				
Owner				
Status	As-Constructed	As-Constructed	As-Constructed	As-Constructed
Notes				
Supporting File				
Type	Bitumen	Bitumen	Bitumen	Bitumen
Length(m)	87.788	127.719	72.756	25.983
Pavement Extension(mm)	1	1	1	1

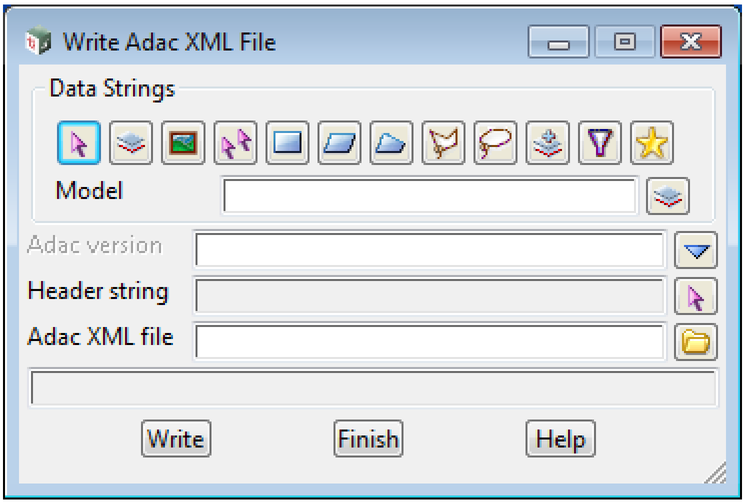
Continue to the next section 1.4.6 Write ADAC XML File or return to 1.4 12d ADAC Menu .

1.4.6 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 1.4.7 Import ADAC XML File or return to 1.4 12d ADAC Menu .

1.4.7 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*

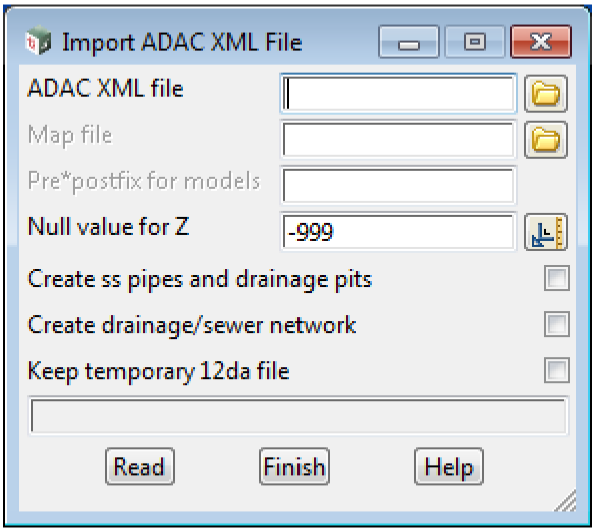
Fitting - in all three of *Sewerage*, *WaterSupply* and *StormWater*

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:


Import ADAC XML File

ADAC XML file

Map file

Pre*postfix for models

Null value for Z -999

Create ss pipes and drainage pits □

Create drainage/sewer network □

Keep temporary 12da file □

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

Field Description	Type	Defaults	Pop-Up
ADAC XML file to import	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*

Map file	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 8.8.1 Create/Edit a Map File *for information about 12d map files.*

Pre*postfix for models	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.*

Null value for Z	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.*

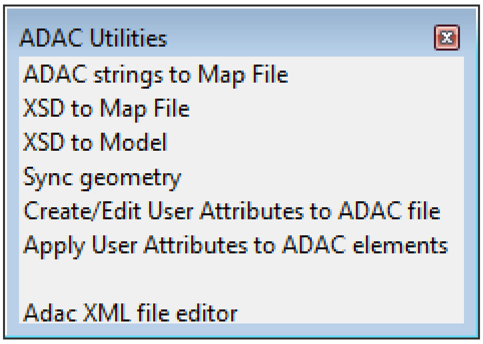
Import button

when clicked, the ADAC XML file is processed and the ADAC data imported into 12d Model.

Continue to the next section 1.4.8 ADAC Utilities or return to 1.4 12d ADAC Menu .

1.4.8 ADAC Utilities

Position of menu: **File I/O =>ADAC =>Utilities**



See

- 1.4.8.1 ADAC Strings to Map File
- 1.4.8.2 XSD to Map File
- 1.4.8.3 XSD to Model
- 1.4.8.4 Sync Geometry
- 1.4.8.5 Create/Edit User Attributes to ADAC File
- 1.4.8.6 Apply User Attributes to ADAC Elements
- 1.4.8.7 ADAC XML File Editor

1.4.8.1 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 **sewertype** 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 **sewertype** 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 **sewertype** 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 **sewertype** 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

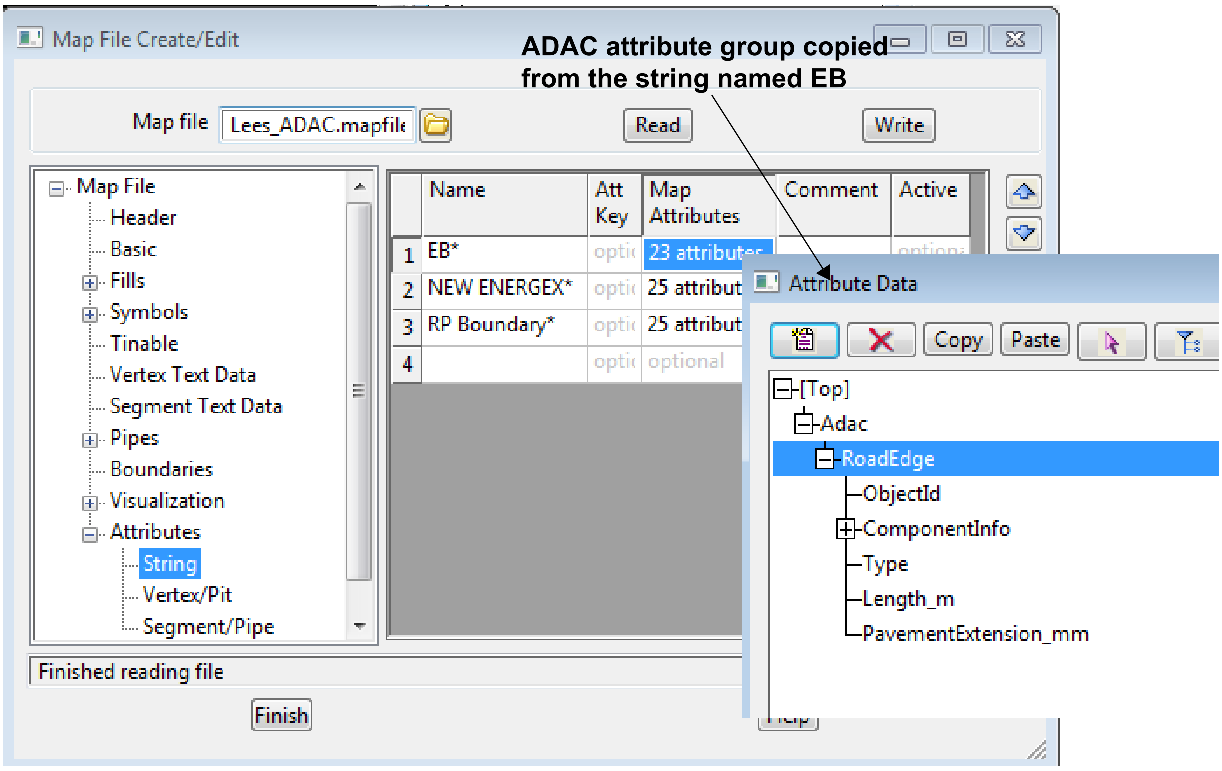
(a) a Road Edge with the string name EB

(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:



ADAC attribute group copied from the string named EB

Map File Create/Edit

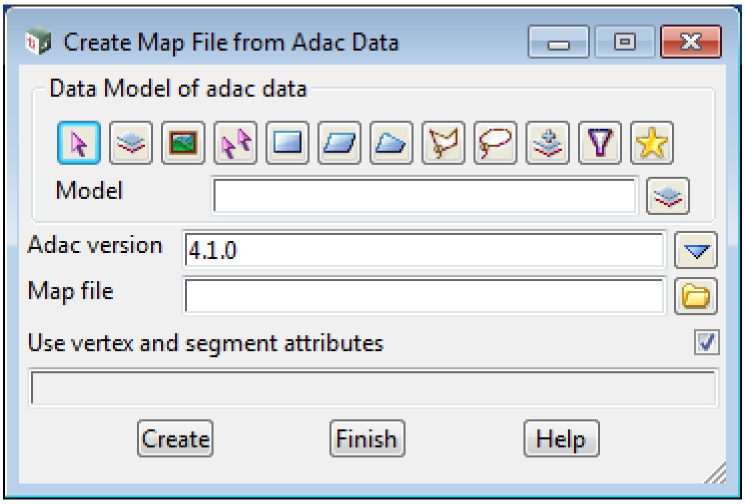
Map file Lees_ADAC.mapfile

	Name	Att Key	Map Attributes	Comment	Active
1	EB*	optic	23 attributes		optional
2	NEW ENERGEX*	optic	25 attribut		
3	RP Boundary*	optic	25 attribut		
4		optic	optional		

Finished reading file

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:



Create Map File from Adac Data

Data Model of adac data

Model

Adac version 4.1.0

Map file

Use vertex and segment attributes ☑

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 create a 12d Map File from.

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 8.8.1 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.*

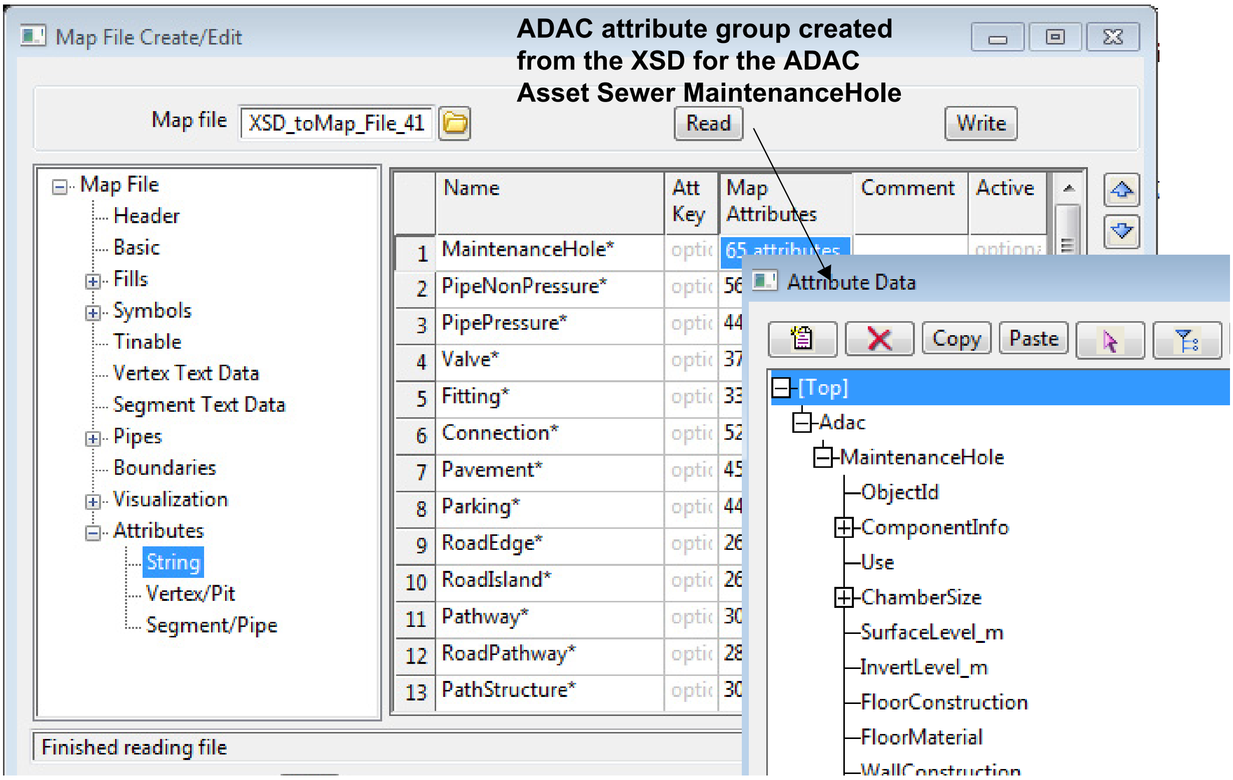
Continue to the next section 1.4.8.2 XSD to Map File or return to 1.4.8 ADAC Utilities .

1.4.8.2 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**:



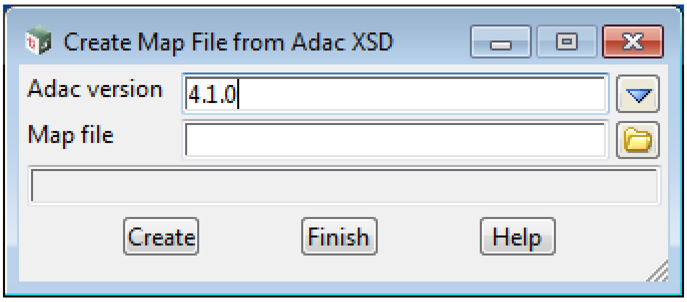
ADAC attribute group created from the XSD for the ADAC Asset Sewer MaintenanceHole

Map file XSD_toMap_File_41

	Name	Att Key	Map Attributes	Comment	Active
1	MaintenanceHole*	optic	65 attributes		optiona
2	PipeNonPressure*	optic	56		
3	PipePressure*	optic	44		
4	Valve*	optic	37		
5	Fitting*	optic	33		
6	Connection*	optic	52		
7	Pavement*	optic	45		
8	Parking*	optic	44		
9	RoadEdge*	optic	26		
10	RoadIsland*	optic	26		
11	Pathway*	optic	30		
12	RoadPathway*	optic	28		
13	PathStructure*	optic	30		

Finished reading file

Selecting **XSD to Map file** brings up the **Create Map File from ADAC XSD** panel:



Adac version 4.1.0

Map file

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

Field Description	Type	Defaults	Pop-Up
ADAC version	choice box	from env.4d	4.0.0, 4.1.0

this is the ADAC XML XSD to create the 12d Map File from.

Map file file box *.mapfile, *.mf files

the name of the ***Map File*** *that is created. And in the* ***Attributes >String*** *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* ***Name*** *column, and in the same row, an ADAC group is created in the* ***Map Attributes*** *column of the type, and all the elements in the XSD for that ADAC Asset are copied as attributes into the ADAC group.*

See the section 8.8.1 Create/Edit a Map File *for information about 12d map files.*

Create button

when clicked, the ***Map File*** *is created.*

Continue to the next section 1.4.8.3 XSD to Model or return to 1.4.8 ADAC Utilities .

1.4.8.3 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

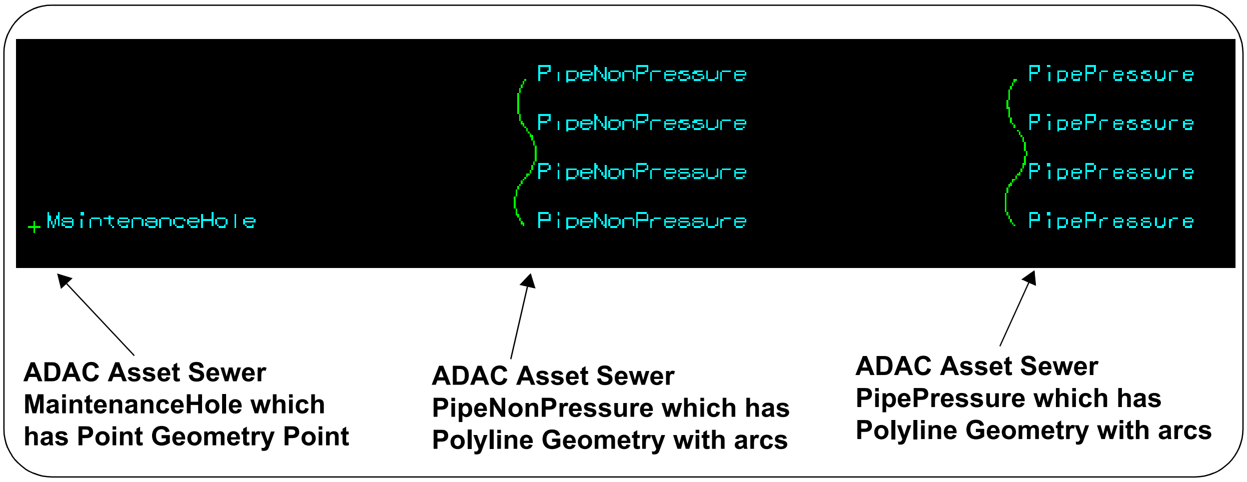
(b) in the string attributes, an ADAC group with default values for each of the ADAC elements

(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:

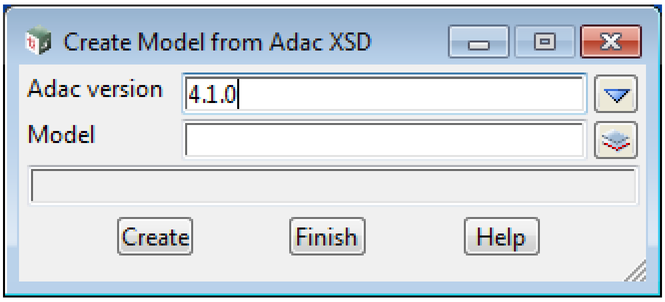


ADAC Asset Sewer MaintenanceHole which has Point Geometry Point

ADAC Asset Sewer PipeNonPressure which has Polyline Geometry with arcs

ADAC Asset Sewer PipePressure which has Polyline Geometry with arcs

Selecting **XSD to model** brings up the **Create Model from ADAC XSD** panel:


Create Model from Adac XSD

Adac version 4.1.0

Model

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

Field Description	Type	Defaults	Pop-Up
ADAC version	choice box	from env.4d	4.0.0, 4.1.0

this is the ADAC XML XSD to create the strings for.

Field Description	Type	Defaults	Pop-Up
Model	model box		available models

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 8.8.1 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.

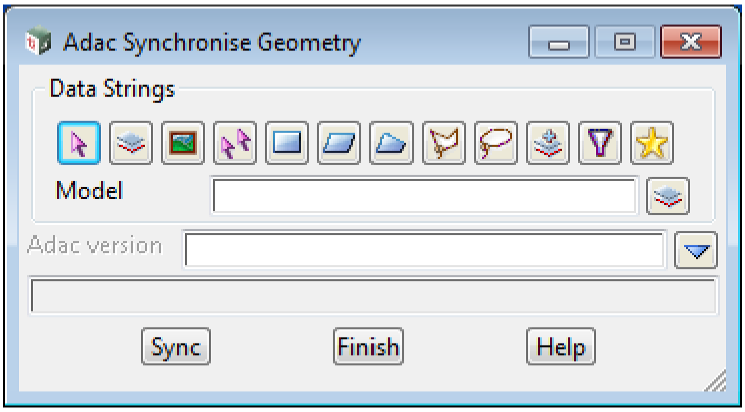
Continue to the next section 1.4.8.4 Sync Geometry or return to 1.4.8 ADAC Utilities .

1.4.8.4 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 1.4.8.5 Create/Edit User Attributes to ADAC File or return to 1.4.8 ADAC Utilities .

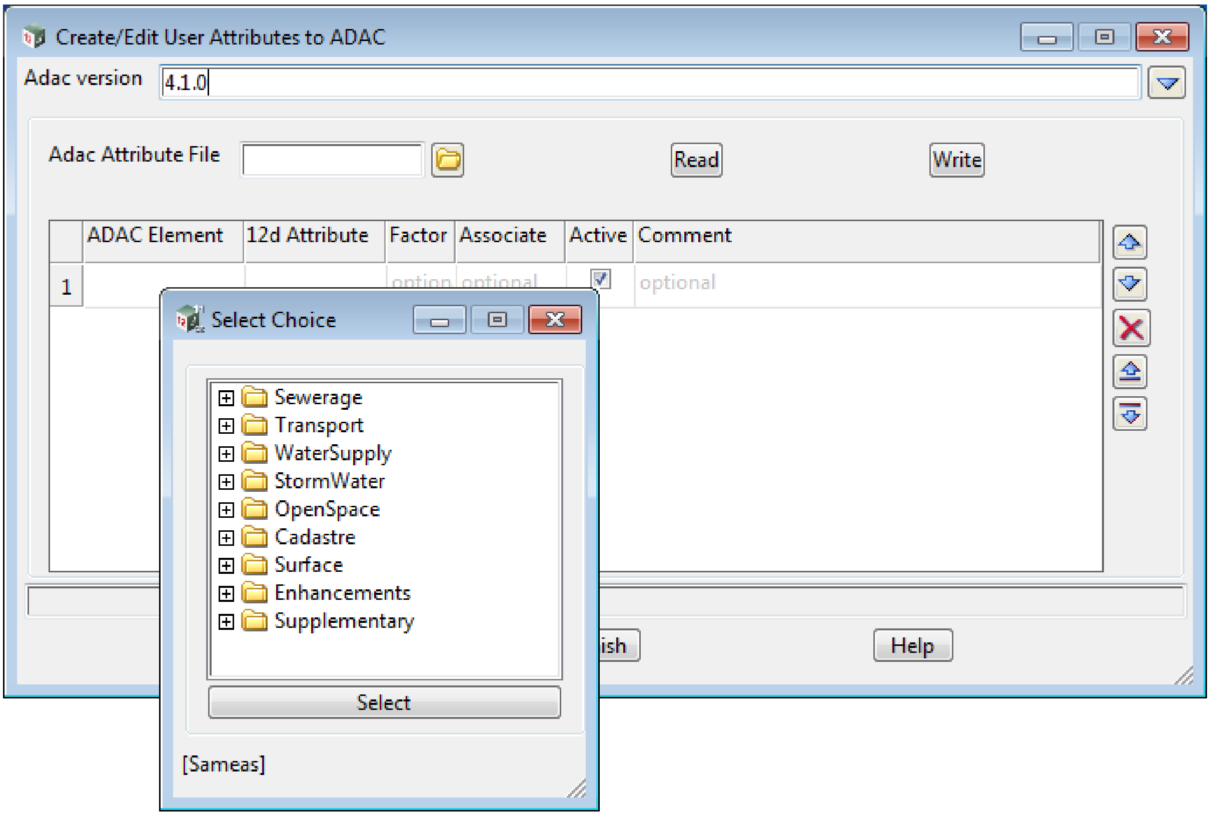
1.4.8.5 Create/Edit User Attributes to ADAC File

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 1.4.8.6 Apply User Attributes to ADAC Elements).

Selecting **Create/Edit User Attributes to ADAC file** brings up the **Create/Edit User Attributes to ADAC** panel:


Create/Edit User Attributes to ADAC

Adac version: 4.1.0

Adac Attribute File:

	ADAC Element	12d Attribute	Factor	Associate	Active	Comment
1			option	optional	☑	optional

Select Choice

- ⊞ Sewerage
- ⊞ Transport
- ⊞ WaterSupply
- ⊞ StormWater
- ⊞ OpenSpace
- ⊞ Cadastre
- ⊞ Surface
- ⊞ Enhancements
- ⊞ Supplementary

[Sameas]

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

Field Description	Type	Defaults	Pop-Up
ADAC version	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.

Field Description	Type	Defaults	Pop-Up
ADAC Attributes file	file box		*.12duaf files

name of the 12d User Attribute to ADAC file to read or write.

Read button

click on this button and the file given in the **ADAC Attribute file** *field is read in and loaded into the editor.*

Write 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.

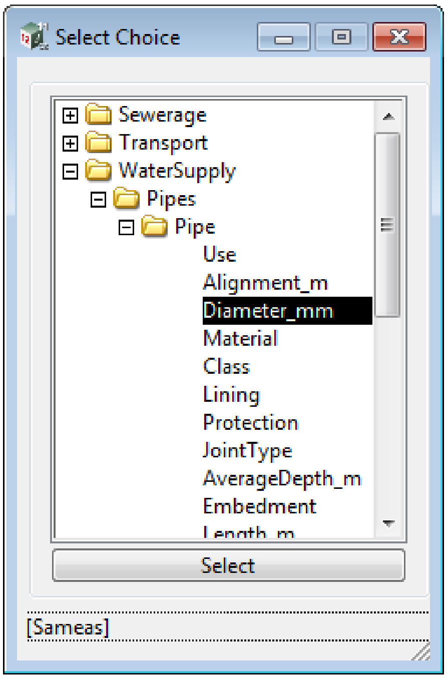
ADAC Element text column

the full name of the ADAC element to have its value updated.

For example,

WaterSupply/Pipes/Pipe/Diameter_mm

*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

*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.*

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.*

Active tick box tick

*if **ticked**, use this row of the grid.*
*If **not ticked**, do not use this row of the grid.*

Comment text

a comment for this row of the grid

Buttons at Bottom

Validate button

when clicked, check that the ADAC Element names against the appropriate ADAC Schema.

Continue to the next section 1.4.8.6 Apply User Attributes to ADAC Elements or return to 1.4.8 ADAC Utilities .

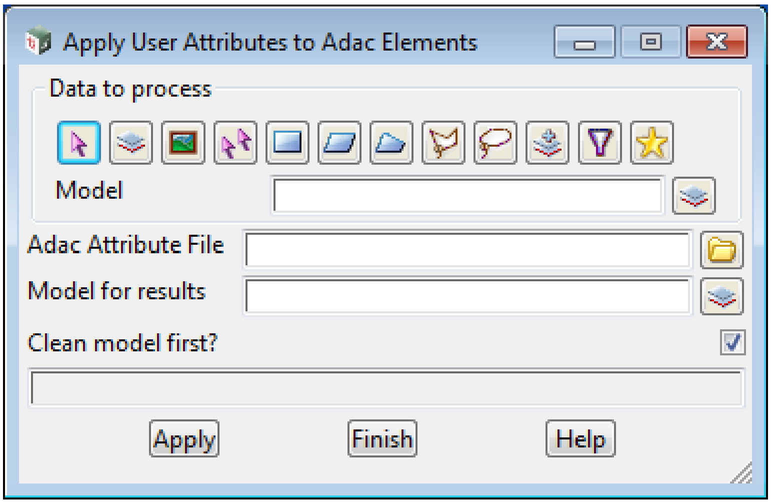
1.4.8.6 Apply User Attributes to ADAC Elements

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 1.4.8.5 Create/Edit User Attributes to ADAC File).

Selecting **Apply User Attributes to ADAC elements** brings up the **Apply User Attributes to ADAC Elements** panel:



Apply User Attributes to Adac Elements

Data to process

Model

Adac Attribute File

Model for results

Clean model first? ☑

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 the ADAC Elements updated by User Attributes.

ADAC Attributes file file box *.12duaf files

name of the 12d User Attribute to ADAC file 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.*

Apply button

when clicked, the selected data will have any ADAC Elements updated by User Attributes according to the selected 12duaf file.

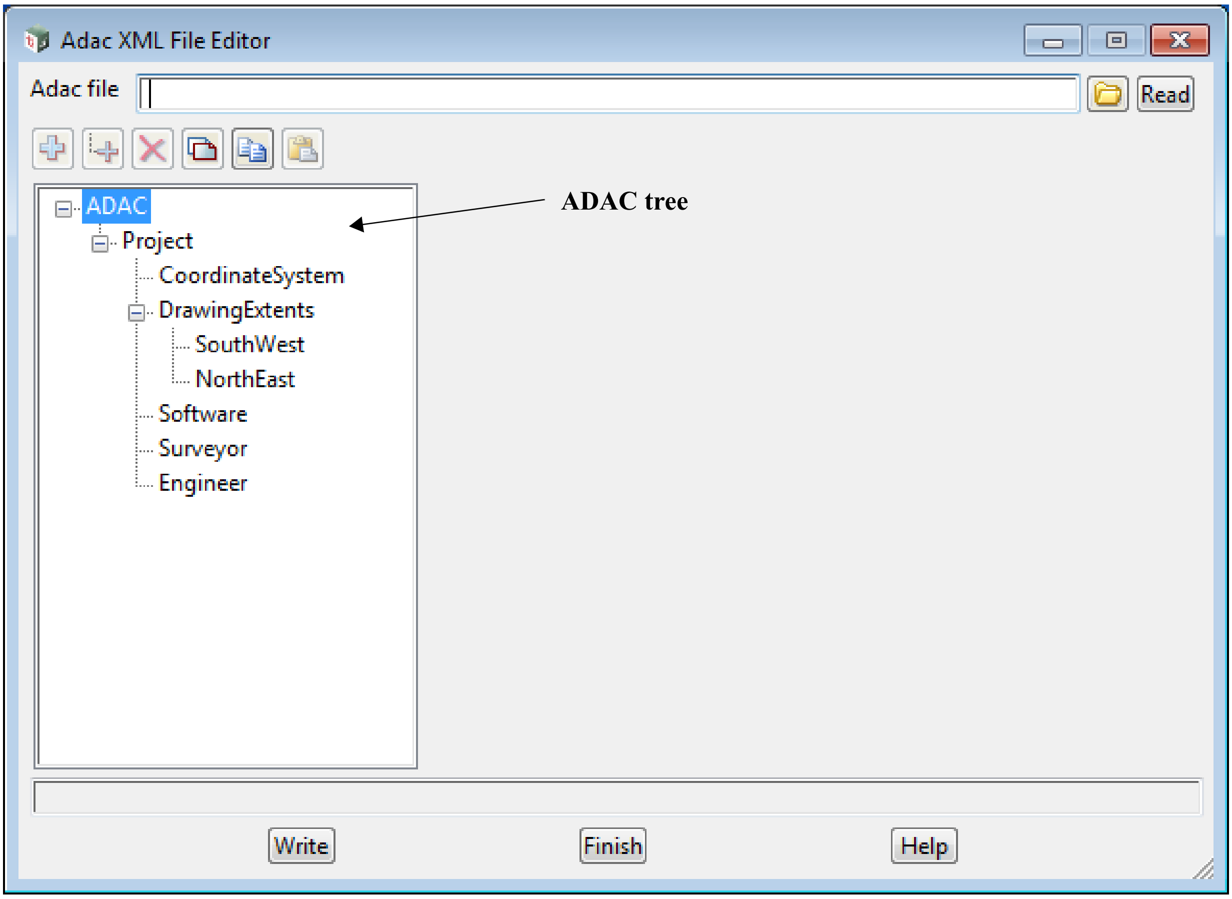
Continue to the next section 1.4.8.7 ADAC XML File Editor or return to 1.4.8 ADAC Utilities .

1.4.8.7 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
ADAC XML file	file box		*.XML files

name of the XML file to read or write.

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

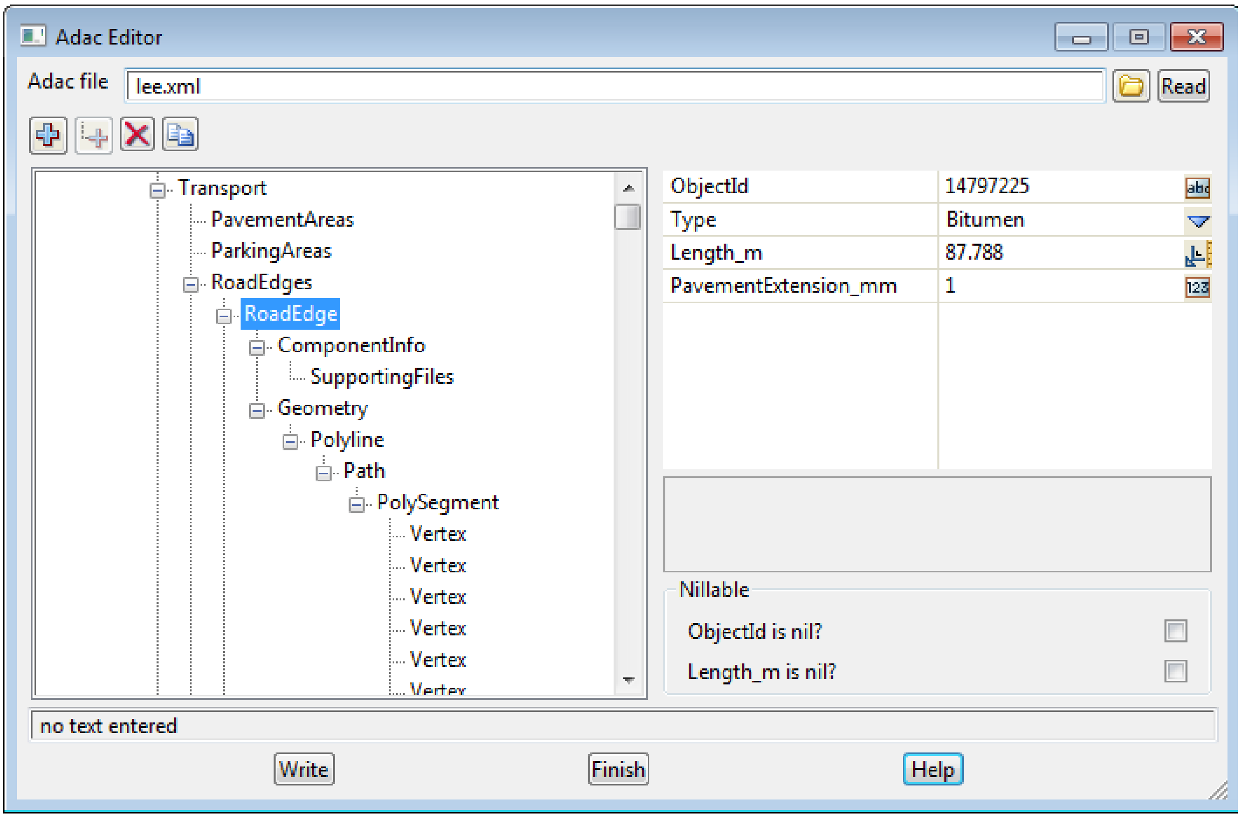
when an ADAC XML file is read in, it is displayed in the ADAC tree.

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,



Adac file lee.xml

ObjectId	14797225
Type	Bitumen
Length_m	87.788
PavementExtension_mm	1

Nillable

ObjectId is nil? □

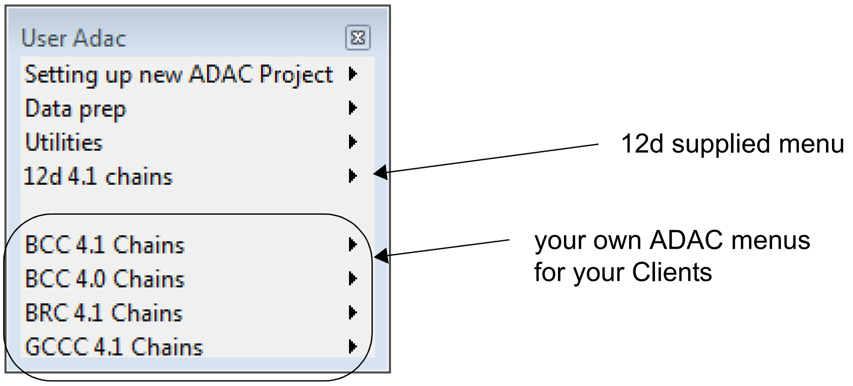
Length_m is nil? □

no text entered

Return to 1.4.8 ADAC Utilities .

1.4.9 User ADAC

Position of menu: **File I/O =>ADAC =>User**



See

1.4.9.1 Setting Up a New ADAC Project
1.4.9.2 Data Prep
1.4.9.3 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 1.4.9.4 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 1.4.9.5 Client Specific ADAC Design & Survey Menus .

Or return to 1.4.8 ADAC Utilities .

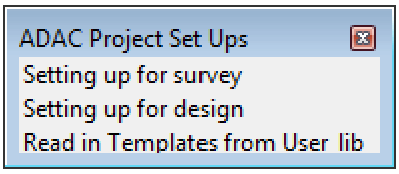
1.4.9.1 Setting Up a New ADAC Project

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

- 1.4.9.1.1 Setting Up for Survey
- 1.4.9.1.2 Setting Up for Design
- 1.4.9.1.3 Reading in Templates from User_Lib

1.4.9.1.1 Setting Up for Survey

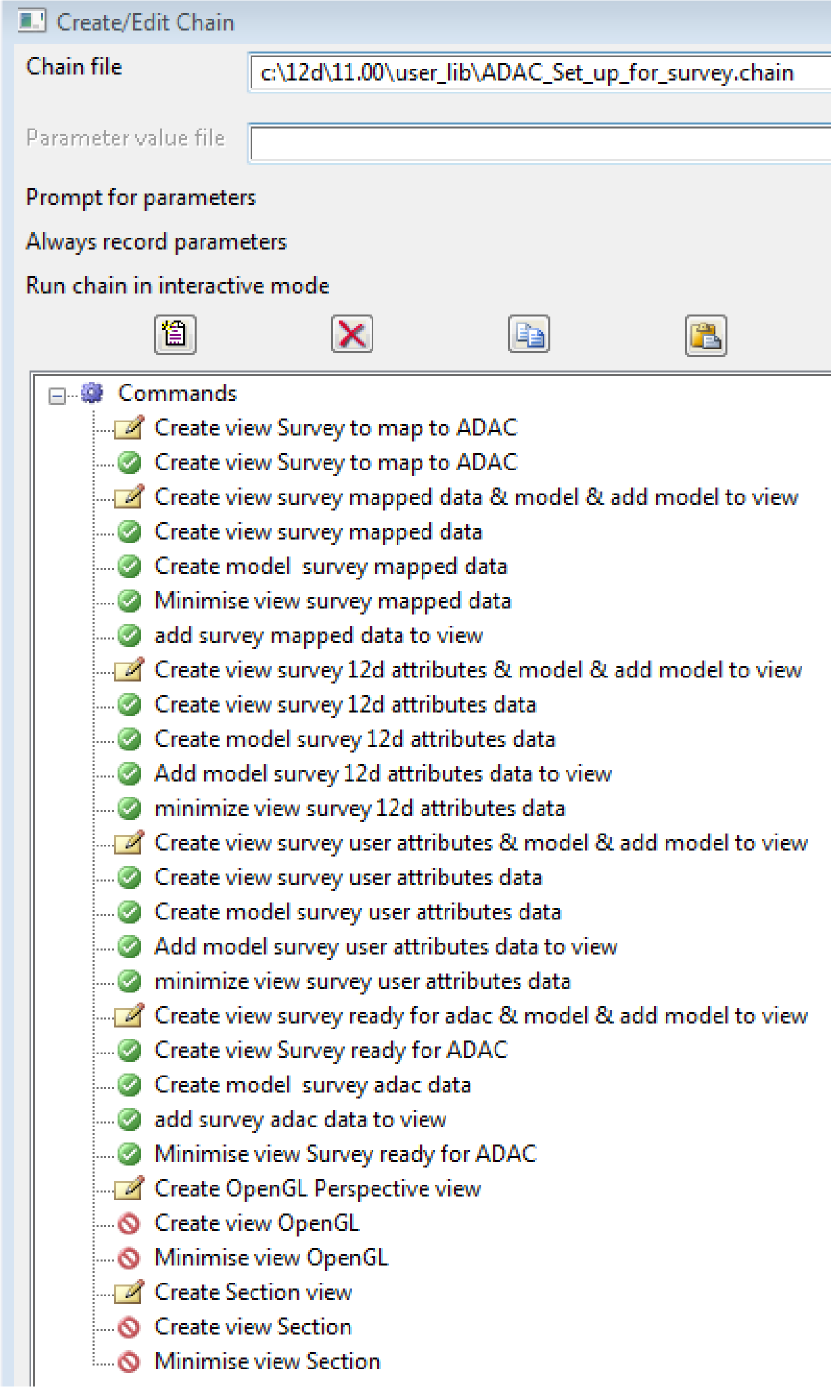
Position of option on menu:

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.



Create/Edit Chain

Chain file c:\12d\11.00\user_lib\ADAC_Set_up_for_survey.chain

Parameter value file

Prompt for parameters

Always record parameters

Run chain in interactive mode

- Commands
  - Create view Survey to map to ADAC
  - Create view Survey to map to ADAC
  - Create view survey mapped data & model & add model to view
  - Create view survey mapped data
  - Create model survey mapped data
  - Minimise view survey mapped data
  - add survey mapped data to view
  - Create view survey 12d attributes & model & add model to view
  - Create view survey 12d attributes data
  - Create model survey 12d attributes data
  - Add model survey 12d attributes data to view
  - minimize view survey 12d attributes data
  - Create view survey user attributes & model & add model to view
  - Create view survey user attributes data
  - Create model survey user attributes data
  - Add model survey user attributes data to view
  - minimize view survey user attributes data
  - Create view survey ready for adac & model & add model to view
  - Create view Survey ready for ADAC
  - Create model survey adac data
  - add survey adac data to view
  - Minimise view Survey ready for ADAC
  - Create OpenGL Perspective view
  - Create view OpenGL
  - Minimise view OpenGL
  - Create Section view
  - Create view Section
  - Minimise view Section

Continue to the next section 1.4.9.1.2 Setting Up for Design or return to 1.4.9.1 Setting Up a New ADAC Project .

1.4.9.1.2 Setting Up for Design

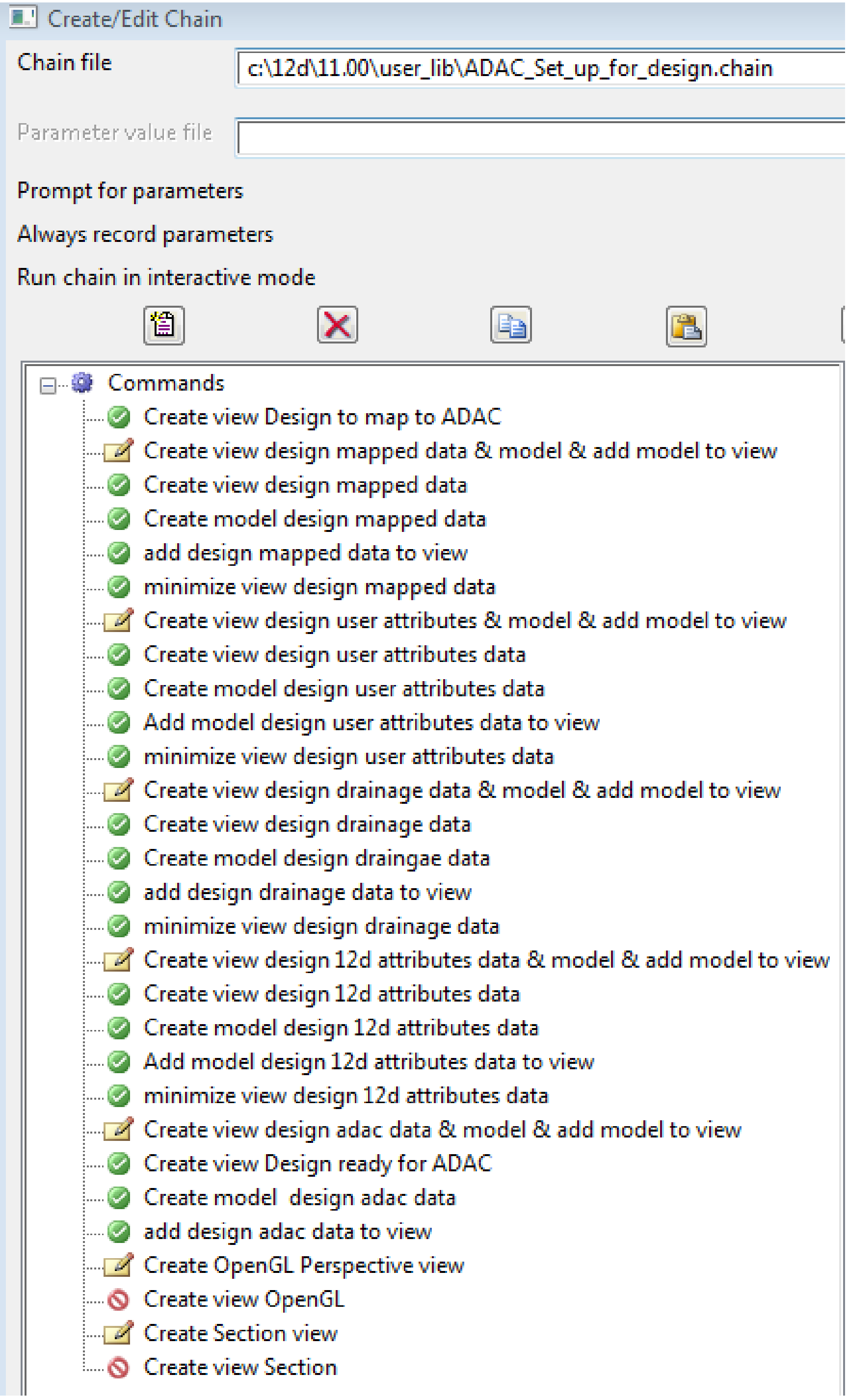
Position of option on menu:

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.



Create/Edit Chain

Chain file: c:\12d\11.00\user_lib\ADAC_Set_up_for_design.chain

Parameter value file

Prompt for parameters

Always record parameters

Run chain in interactive mode

Commands
- ✓ Create view Design to map to ADAC
- Create view design mapped data & model & add model to view
- ✓ Create view design mapped data
- ✓ Create model design mapped data
- ✓ add design mapped data to view
- ✓ minimize view design mapped data
- Create view design user attributes & model & add model to view
- ✓ Create view design user attributes data
- ✓ Create model design user attributes data
- ✓ Add model design user attributes data to view
- ✓ minimize view design user attributes data
- Create view design drainage data & model & add model to view
- ✓ Create view design drainage data
- ✓ Create model design draingae data
- ✓ add design drainage data to view
- ✓ minimize view design drainage data
- Create view design 12d attributes data & model & add model to view
- ✓ Create view design 12d attributes data
- ✓ Create model design 12d attributes data
- ✓ Add model design 12d attributes data to view
- ✓ minimize view design 12d attributes data
- Create view design adac data & model & add model to view
- ✓ Create view Design ready for ADAC
- ✓ Create model design adac data
- ✓ add design adac data to view
- Create OpenGL Perspective view
- ⊘ Create view OpenGL
- Create Section view
- ⊘ Create view Section

Continue to the next section 1.4.9.1.3 Reading in Templates from User_Lib or return to 1.4.9.1 Setting Up a New ADAC Project .

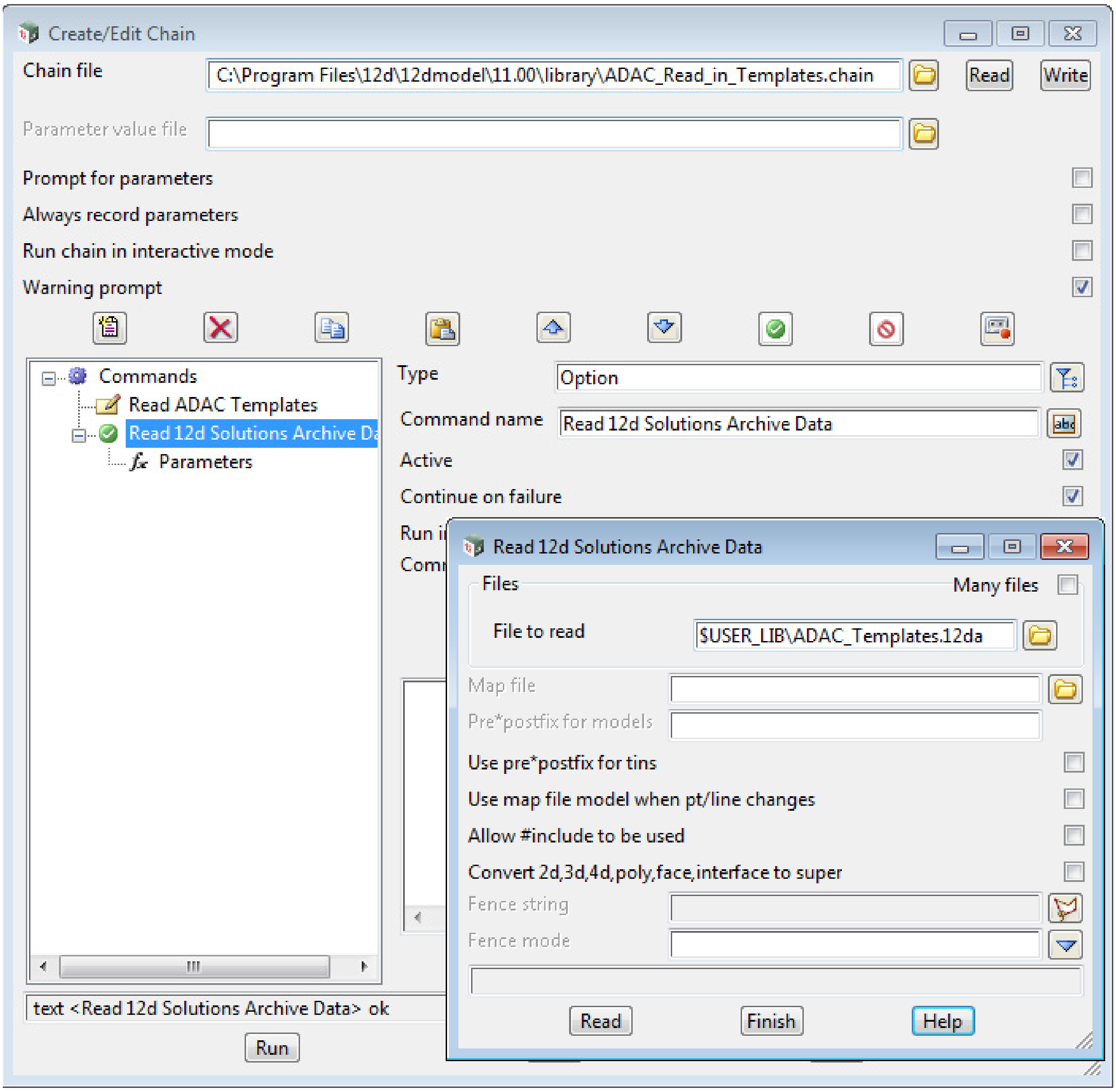
1.4.9.1.3 Reading in Templates from User_Lib

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*.



Create/Edit Chain

Chain file: C:\Program Files\12d\12dmodel\11.00\library\ADAC_Read_in_Templates.chain

Parameter value file:

Prompt for parameters □

Always record parameters □

Run chain in interactive mode □

Warning prompt ☑

Commands
- Read ADAC Templates
- Read 12d Solutions Archive Data
  - Parameters

Type: Option

Command name: Read 12d Solutions Archive Data

Active ☑

Continue on failure ☑

text <Read 12d Solutions Archive Data> ok

Read 12d Solutions Archive Data

Files — Many files □

File to read: $USER_LIB\ADAC_Templates.12da

Map file:

Pre*postfix for models:

Use pre*postfix for tins □

Use map file model when pt/line changes □

Allow #include to be used □

Convert 2d,3d,4d,poly,face,interface to super □

Fence string:

Fence mode:

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 1.6.4 Setting Up ADAC Templates .

Return to 1.4.9.1 Setting Up a New ADAC Project .

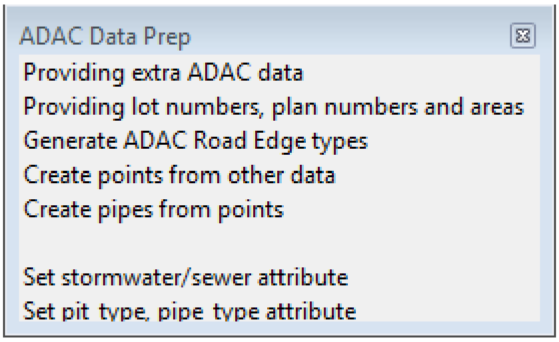
1.4.9.2 Data Prep

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.


ADAC Data Prep
Providing extra ADAC data
Providing lot numbers, plan numbers and areas
Generate ADAC Road Edge types
Create points from other data
Create pipes from points
Set stormwater/sewer attribute
Set pit type, pipe type attribute

See

- 1.4.9.2.1 Providing Extra Data for ADAC
- 1.4.9.2.2 Providing Lot and Plan Numbers and Areas for ADAC
- 1.4.9.2.3 Generate ADAC Road Edge Types
- 1.4.9.2.4 Create Points from Other Data
- 1.4.9.2.5 Create Pipes from Points
- 1.4.9.2.6 Set Stormwater/Sewer Property
- 1.4.9.2.7 Set a Drainage-Sewer Pit and Pipe Type Attribute

1.4.9.2.1 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 (1.5.1.2.4 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.

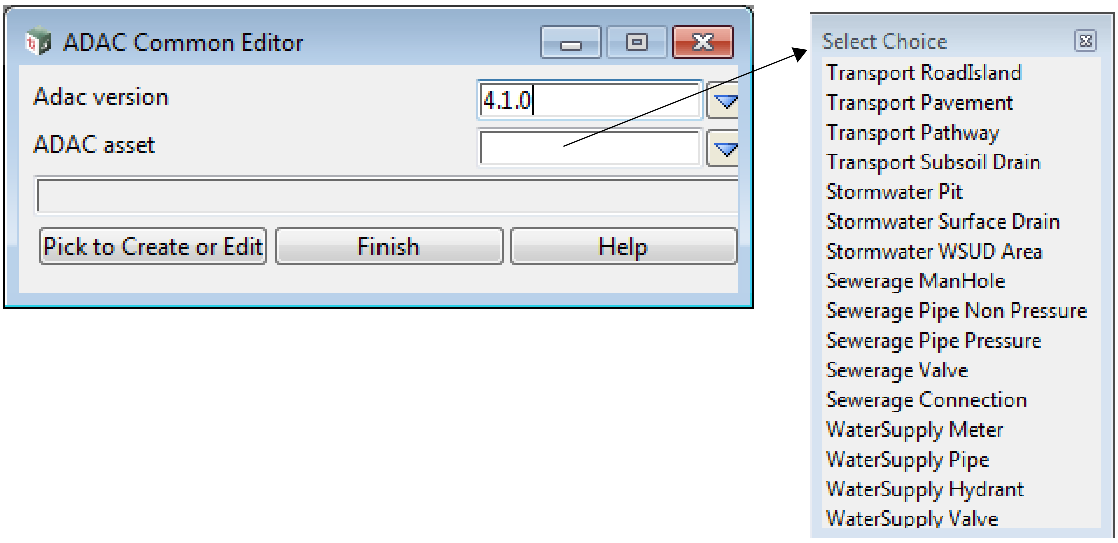
Another reason is that at this stage the *ADAC Survey* and *Design* chains have not been run to mark strings as ADAC Assets and set up the ADAC attributes group.

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



ADAC Common Editor

Adac version 4.1.0

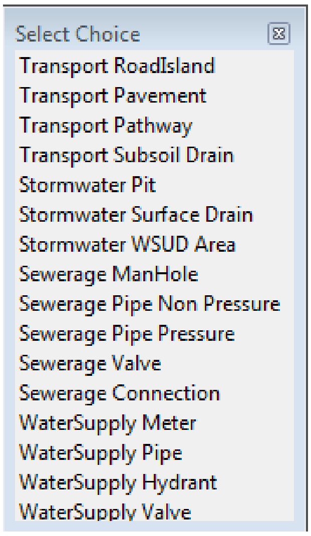
ADAC asset

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

Field Description	Type	Defaults	Pop-Up
ADAC version	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


Select Choice
Transport RoadIsland
Transport Pavement
Transport Pathway
Transport Subsoil Drain
Stormwater Pit
Stormwater Surface Drain
Stormwater WSUD Area
Sewerage ManHole
Sewerage Pipe Non Pressure
Sewerage Pipe Pressure
Sewerage Valve
Sewerage Connection
WaterSupply Meter
WaterSupply Pipe
WaterSupply Hydrant
WaterSupply Valve

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.

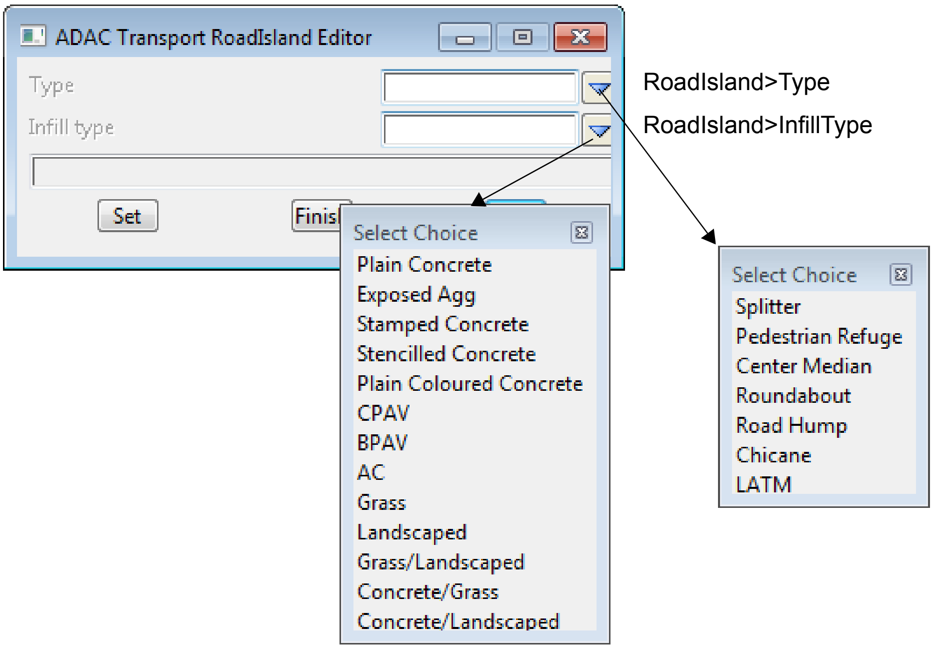
If the picked string has not yet had attributes set by this option AND **ADAC asset** *has a value, then a panel with the attributes that this option can set for the given* **ADAC asset** *is displayed and then created for the picked string.*

1.4.9.2.1.1 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.

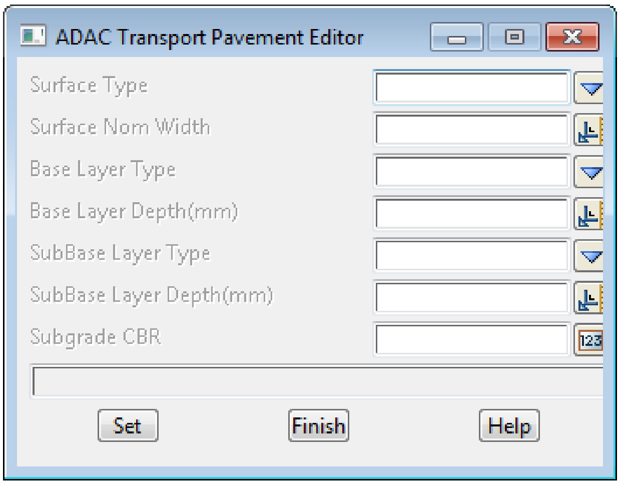
Continue to the next section 1.4.9.2.1.2 Transport Pavement or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.2 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.



Surface>SurfaceType

Surface>SurfaceNomWidth_m

PavementStructure>BaseLayer>LayerType

PavementStructure>BaseLayer>LayerDepth_mm

PavementStructure>SubBaseLayer>LayerType

PavementStructure>SubBaseLayer>LayerDepth_mm

Subgrade>CBR

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.

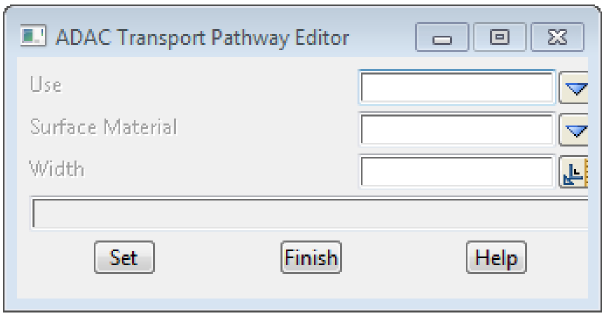
Continue to the next section 1.4.9.2.1.3 Transport Pathway or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.3 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.



Pathway>Use

Pathway>SurfaceMaterial

Pathway>Width_m

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.

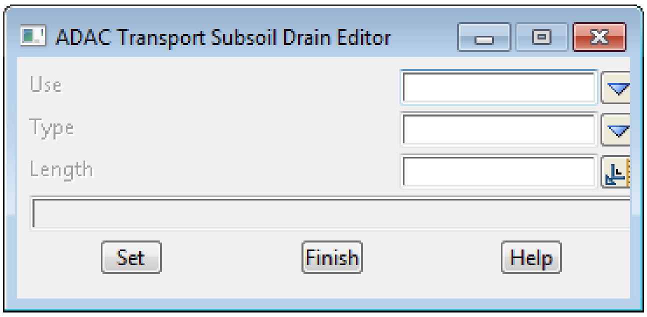
Continue to the next section 1.4.9.2.1.4 Transport Subsoil Drain or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.4 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.



SubSoilDrain>Use

SubSoilDrain>Type

SubSoilDrain>Lenght_m

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.

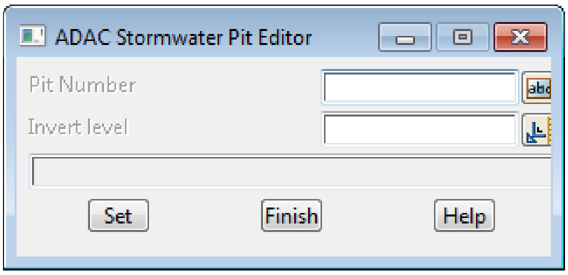
Continue to the next section 1.4.9.2.1.5 Stormwater Pit or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.5 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.



StormWater>Pits>PitNumber

StormWater>Pits>dimensions_point_levels_m>InvertLevel_m

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.

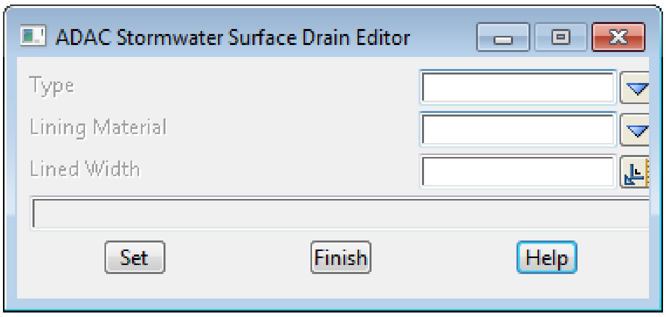
Continue to the next section 1.4.9.2.1.6 Stormwater Surface Drain or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.6 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.



StormWater>SubSurfaceDrain>Type

StormWater>SubSurfaceDrain>LiningMaterial

StormWater>SubSurfaceDrain>LinedWidth_m

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.

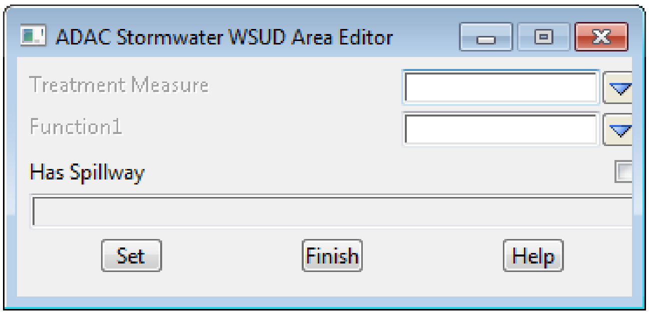
Continue to the next section 1.4.9.2.1.7 Stormwater WSUD Area or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.7 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.



StormWater>WSUDArea>TreatmentMeasure

StormWater>WSUDArea>Function1

StormWater>WSUDArea>HasSpillway

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.

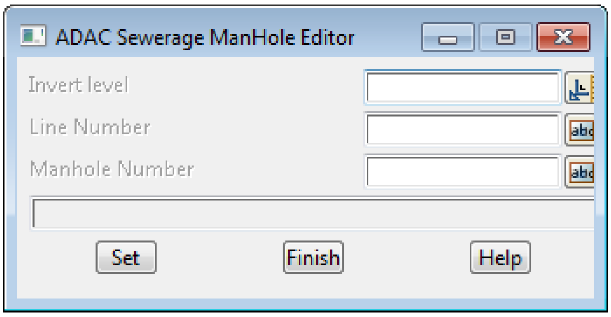
Continue to the next section 1.4.9.2.1.8 Sewerage Maintenance Hole or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.8 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.



MaintenanceHole>InvertLevel_m

MaintenanceHole>LineNumber

MaintenanceHole>MH_Number

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.

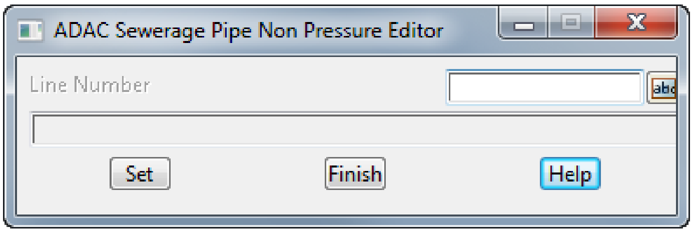
Continue to the next section 1.4.9.2.1.9 Sewerage Pipe Non Pressure or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.9 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.

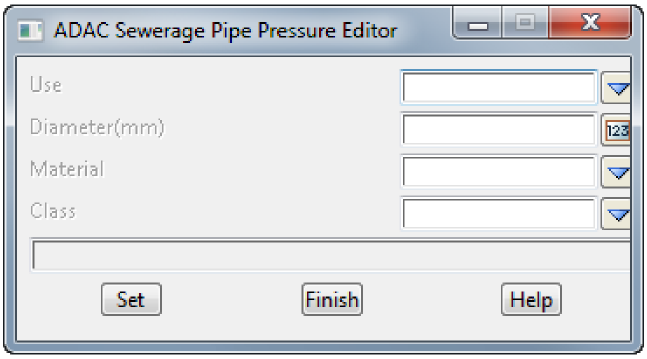
Continue to the next section 1.4.9.2.1.10 Sewerage Pipe Pressure Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.10 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.

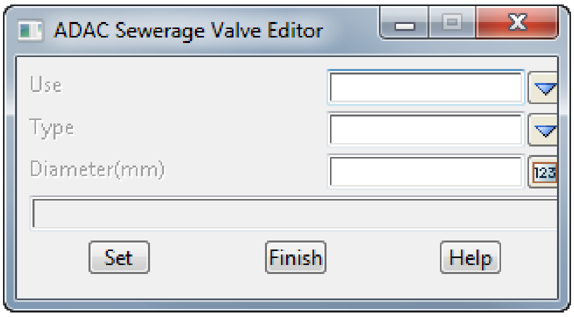
Continue to the next section 1.4.9.2.1.11 Sewerage Valve Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.11 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.



Valve>Use

Valve>Type

Valve>Diameter_mm

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.

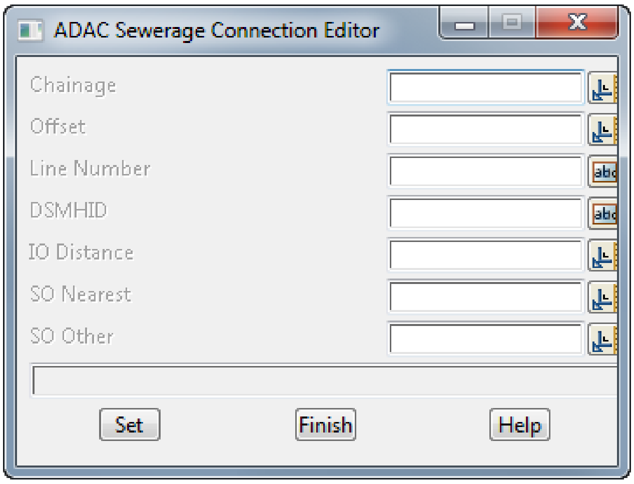
Continue to the next section 1.4.9.2.1.12 Sewerage Connection Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.12 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.



Connection>Chainage_m

Connection>Offset_m

Connection>LineNumber

Connection>DSMHID

Connection>IO_Distance_m

Connection>SO_Nearest_m

Connection>SO_Other_m

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.

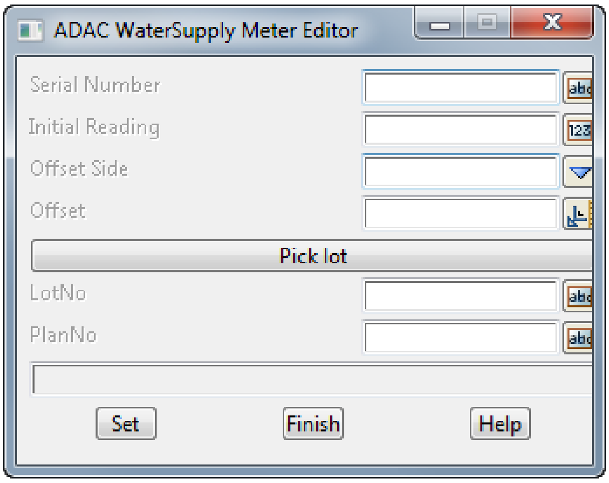
Continue to the next section 1.4.9.2.1.13 Water Supply Meter Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.13 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 1.4.9.2.2 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.

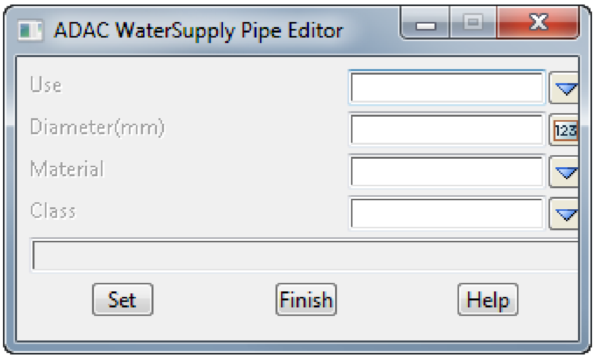
Continue to the next section 1.4.9.2.1.14 Water Supply Pipe Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.14 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.



Pipe>Use

Pipe>Diameter_mm

Pipe>Material

Pipe>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.

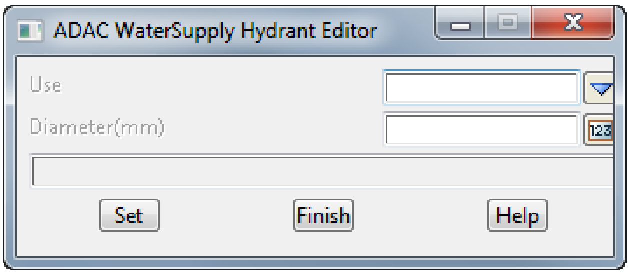
Continue to the next section 1.4.9.2.1.15 Water Supply Hydrant Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.15 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.



Hydrant>Use

Hydrant>Diameter_mm

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.

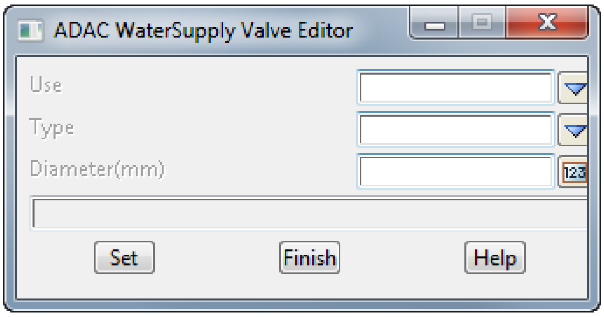
Continue to the next section 1.4.9.2.1.16 Water Supply Valve Editor or return to 1.4.9.2.1 Providing Extra Data for ADAC .

1.4.9.2.1.16 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.



Valve>Use

Valve>Type

Valve>Diameter_mm

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 1.4.9.2.2 Providing Lot and Plan Numbers and Areas for ADAC or return to 1.4.9.2 Data Prep .

1.4.9.2.2 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 (1.5.1.2.4 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.

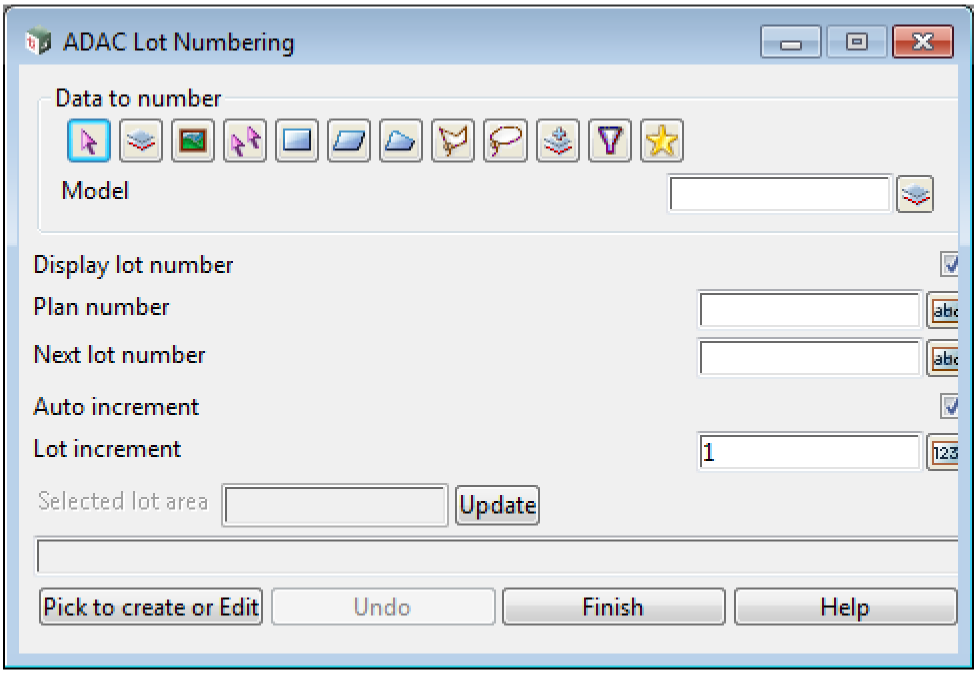
Another reason is that at this stage the *ADAC Survey* and *Design* chains have not been run to mark strings as ADAC Assets and set up the ADAC attributes group.

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:



Lot increment: 1

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 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 **Lot increment** *is written to the* **Next lot no.** *field.*

If a Model is selected by MB same as, you need to type <Enter> after it for the search to go ahead.

Display lot number tick box tick

if ***ticked****, the lot numbers for existing lots are temporarily displayed on the view.*

If ***not ticked,*** *the lot numbers are not temporarily displayed on the view.*

Plan number text box

number of the plan the lots are from.

Next lot number text box

the next lot number to use.

Auto increment tick box tick

if ***ticked****, as each lot number is labelled, the* **Next lot number** *is incremented by the* **Lot increment**

If ***not ticked,*** *the* **Next lot number** *field is not incremented.*

Lot increment number box 1

the amount to increment the **Next lot number** *by.*

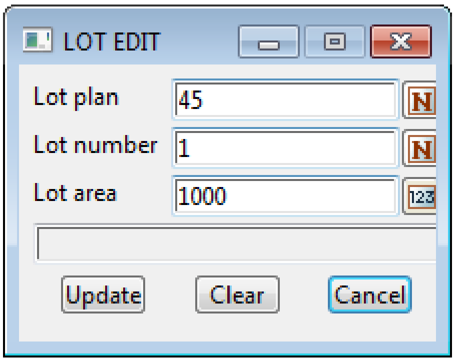
Selected lot area real box

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 **Update**.

Pick to Create or Edit button

click on the button and then pick a string.

If the picked string already has Lot attributes set by this option (or other options) then those values will be displayed in the **Lot Edit** *panel, and modified and saved by clicking* **Update** *on that panel.*



Lot plan 45

Lot number 1

Lot area 1000

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.*

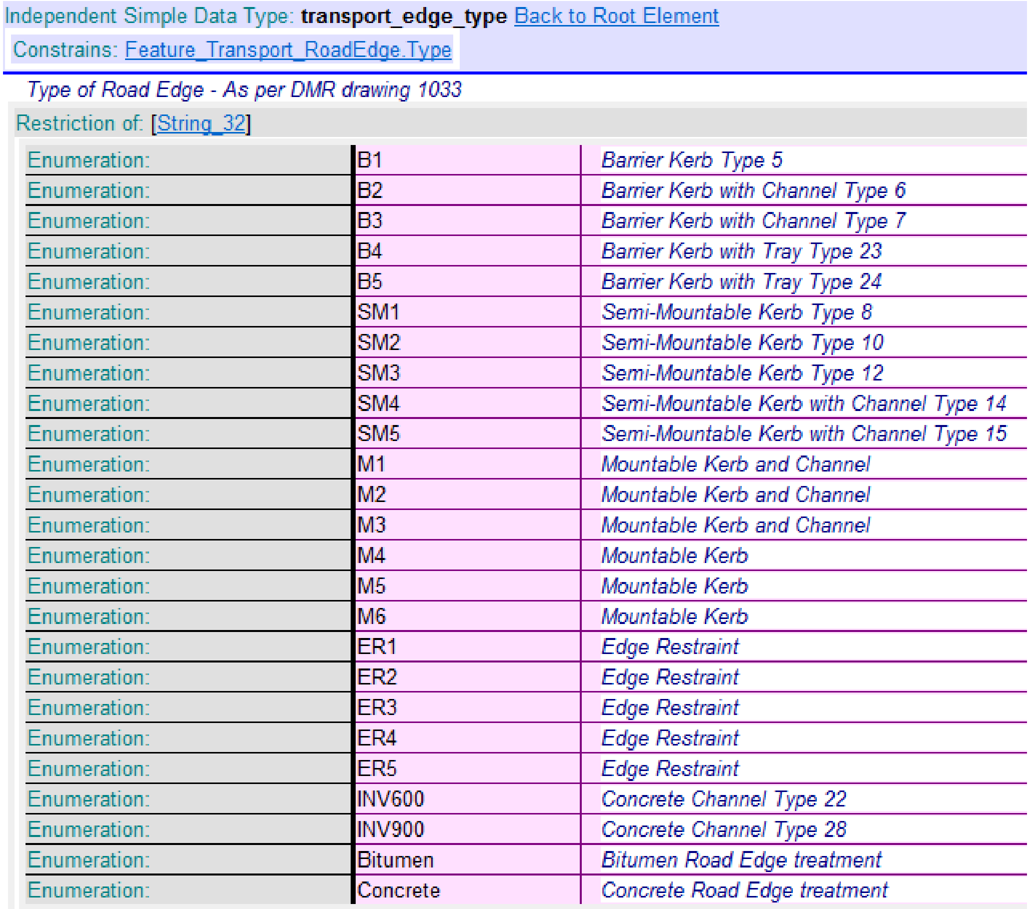
Continue to the next section 1.4.9.2.3 Generate ADAC Road Edge Types or return to 1.4.9.2 Data Prep .

1.4.9.2.3 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*

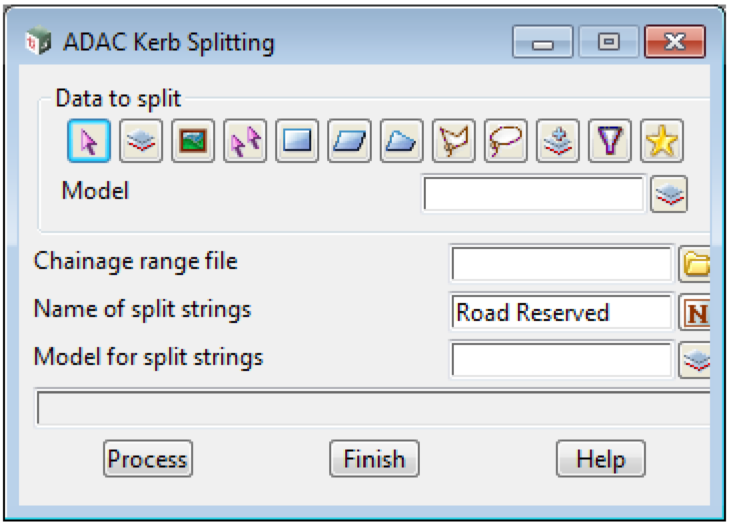
However, many companies do not have a naming convention that differentiates between the choices and may use just the one string name for all of them. So there is no way to automatically know the **Type** which is **mandatory** for an ADAC *Transport>RoadEdges>RoadEdge* Asset.

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



Name of split strings: Road Reserved

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 that is searched for strings to split.

Chainage range file 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

Name of split strings 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.*

Model for split strings model box available models

the model for the created split strings.

Process button

go and create the split strings.

Continue to the next section 1.4.9.2.4 Create Points from Other Data or return to 1.4.9.2 Data Prep .

1.4.9.2.4 Create Points from Other Data

Position of menu:

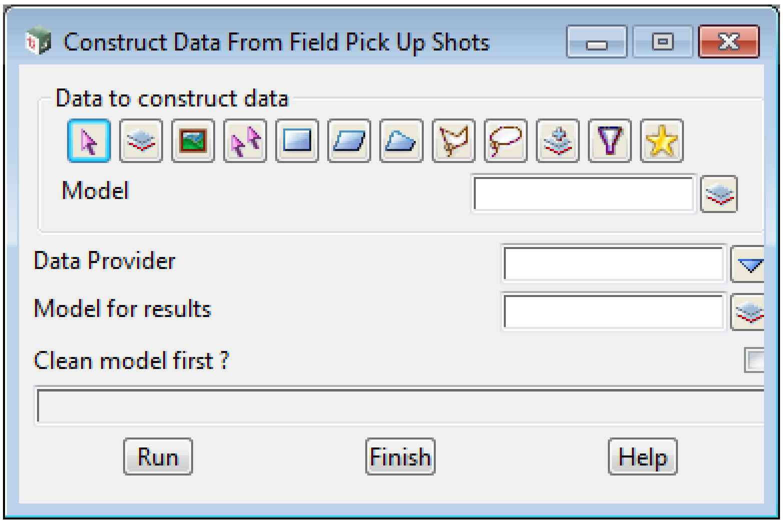
File I/O =>ADAC =>User =>Data prep =>Create points from other data

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

Data source type Model

data selection type - for a full description go to 4.19.3 Data Source .

Data source input

source of strings that is process to generate strings suitable for ADAC.

Data Provider choice box available data providers

selects the table of modifications to make.

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

process the data.

Continue to the next section 1.4.9.2.5 Create Pipes from Points or return to 1.4.9.2 Data Prep .

1.4.9.2.5 Create Pipes from Points

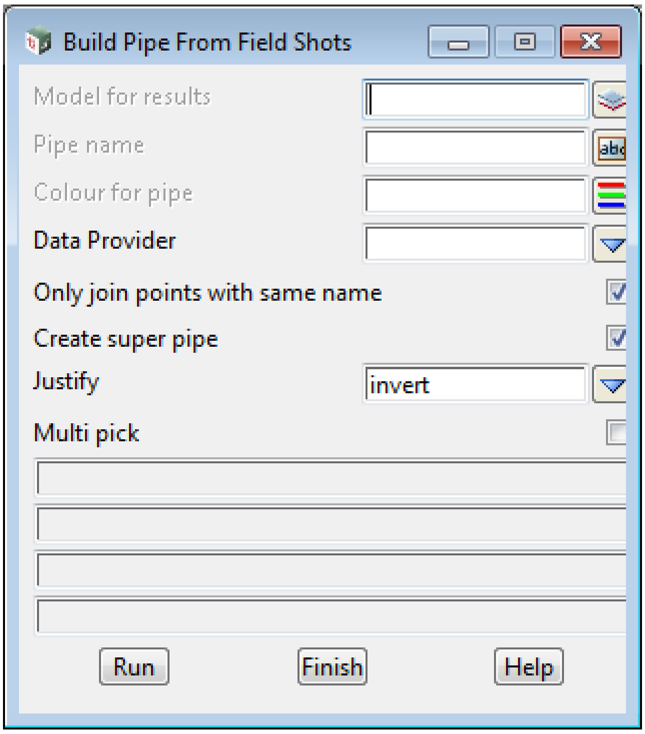
Position of menu:

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



Model for results

Pipe name

Colour for pipe

Data Provider

Only join points with same name ☑

Create super pipe ☑

Justify invert

Multi pick □

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

Field Description	Type	Defaults	Pop-Up
Model for results	model box		available models

model that the create pipes are added to.

Pipe name	text box		

*if **not blank**, the created string is given this name.*
If blank,

Colour for pipe	colour box		available colours

*if **not blank**, the created string is given this colour.*
If blank,

Data Provider	choice box		available data providers

selects the table of attributes to use for diameter etc.

Only join points with same name tick box

*if **ticked**, after picking the first one vertex string, the second one vertex string must have the same name as the first picked string.*

Create super pipe tick box

*if **ticked**, and there is an attribute specified in the selected* **Data provider** *for diameter, then a super string pipe with the given diameter is created.*

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 1.4.9.2.6 Set Stormwater/Sewer Property or return to 1.4.9.2 Data Prep .

1.4.9.2.6 Set Stormwater/Sewer Property

Position of menu:

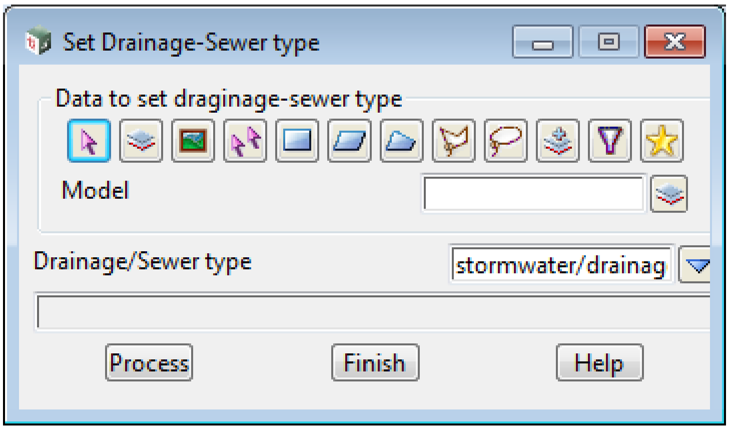
File I/O =>ADAC =>User =>Data prep =>Set stormwater/sewer purpose

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



Drainage/Sewer type: stormwater/drainag

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.

Drainage/Sewer purpose	choice box	stormwater/drainage, wastewater/sewer	

the Purpose property of the selected strings are set to this value.

Run	button		

set the Purpose property of the selected strings.

Continue to the next section 1.4.9.2.7 Set a Drainage-Sewer Pit and Pipe Type Attribute or return to 1.4.9.2 Data Prep .

1.4.9.2.7 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**.

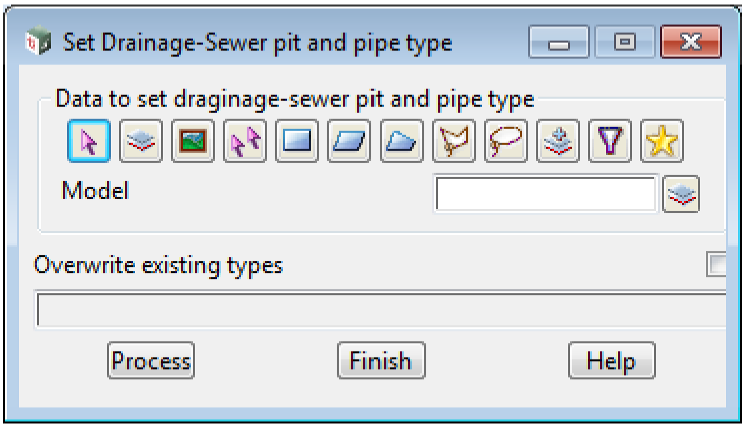
So after using the DNE, the attributes pit_type and pipe_type are available to be used in the ADAC ***Map File***. However if the DNE is not run, the attributes pit_type and pipe_type will not exist.

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


Set Drainage-Sewer pit and pipe type
Data to set dragінage-sewer pit and pipe type
Model
Overwrite existing types
Process
Finish
Help

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.

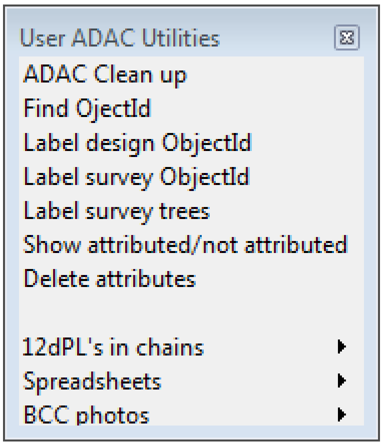
Return to 1.4.9.2 Data Prep or 1.4.9 User ADAC .

1.4.9.3 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

1.4.9.3.1 ADAC Clean Up

1.4.9.3.2 Find Object Id

1.4.9.3.3 Label Design/Survey Object Id

1.4.9.3.4 Show Attributed/Not Attributed

1.4.9.3.5 Delete ADAC Attributes

1.4.9.3.6 12dPL’s in Chains

1.4.9.3.1 ADAC Clean Up

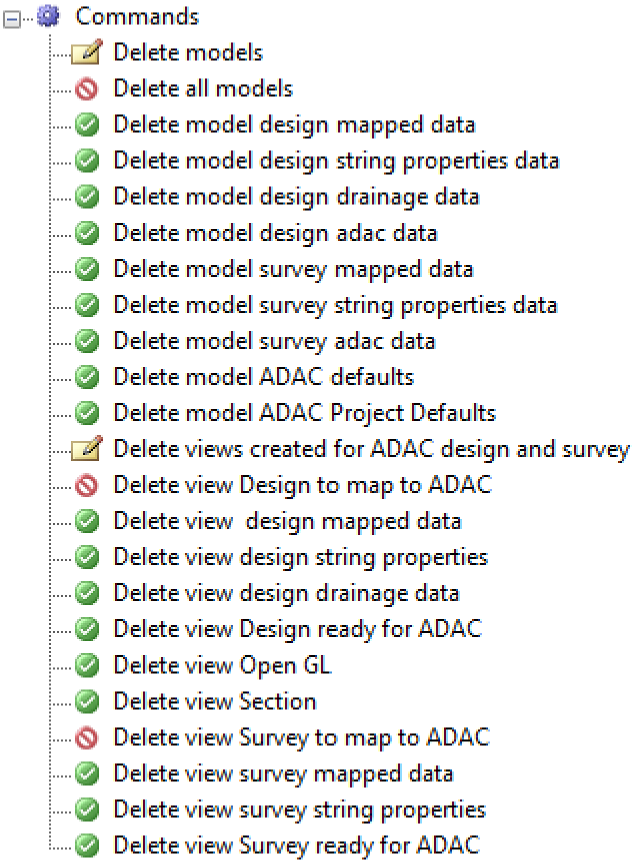
Position of option on menu: **File I/O =>ADAC =>User =>Utilities =>ADAC clean up**

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



Commands
- Delete models
- Delete all models
- Delete model design mapped data
- Delete model design string properties data
- Delete model design drainage data
- Delete model design adac data
- Delete model survey mapped data
- Delete model survey string properties data
- Delete model survey adac data
- Delete model ADAC defaults
- Delete model ADAC Project Defaults
- Delete views created for ADAC design and survey
- Delete view Design to map to ADAC
- Delete view design mapped data
- Delete view design string properties
- Delete view design drainage data
- Delete view Design ready for ADAC
- Delete view Open GL
- Delete view Section
- Delete view Survey to map to ADAC
- Delete view survey mapped data
- Delete view survey string properties
- Delete view Survey ready for ADAC

The chain needs no interaction with the user; it just runs and ends.

Continue to the next section 1.4.9.3.2 Find Object Id or return to 1.4.9.3 User ADAC Utilities .

1.4.9.3.2 Find Object Id

Position of option on menu: **File I/O =>ADAC =>User =>Utilities =>Find Objectid**

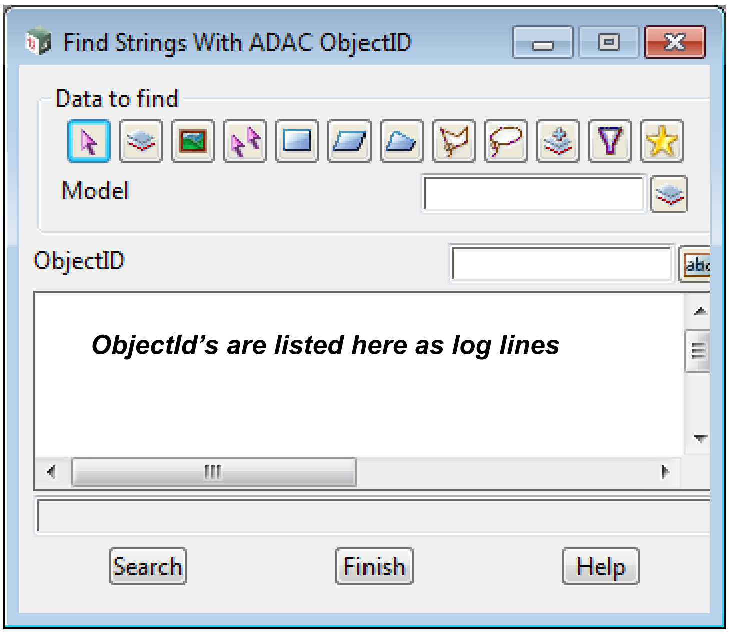
From the *ADAC XML Schema, the* **Objectid**

> *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 1.4.5 Report) and any ADAC XML file that is created (see 1.4.6 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:



ObjectId's are listed here as log lines

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 1.4.9.3.3 Label Design/Survey Object Id or return to 1.4.9.3 User ADAC Utilities .

1.4.9.3.3 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.*

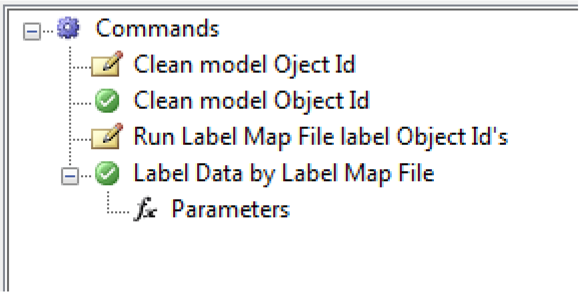
This option runs an Interactive chain that includes a **Label Data by Label Map File** panel that creates labels for all the ADAC assets in the model **design adac data** or **survey adac data** and cleans and adds them to the model **ObjectId**.

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/survevObjectid** runs the Interactive chain

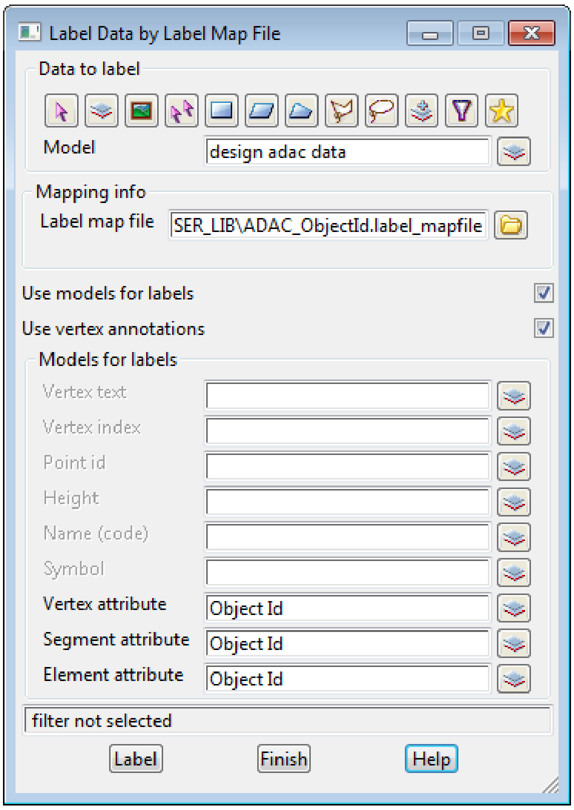
$LIB/ADAC_Label_design_ObjectId.chain

$LIB/ADAC_Label_survey_ObjectId.chain


Commands
Clean model Oject Id
Clean model Object Id
Run Label Map File label Object Id's
Label Data by Label Map File
Parameters

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.



Label Data by Label Map File

Data to label

Model: design adac data

Mapping info

Label map file: SER_LIB\ADAC_ObjectId.label_mapfile

Use models for labels ☑

Use vertex annotations ☑

Models for labels

Vertex text:

Vertex index:

Point id:

Height:

Name (code):

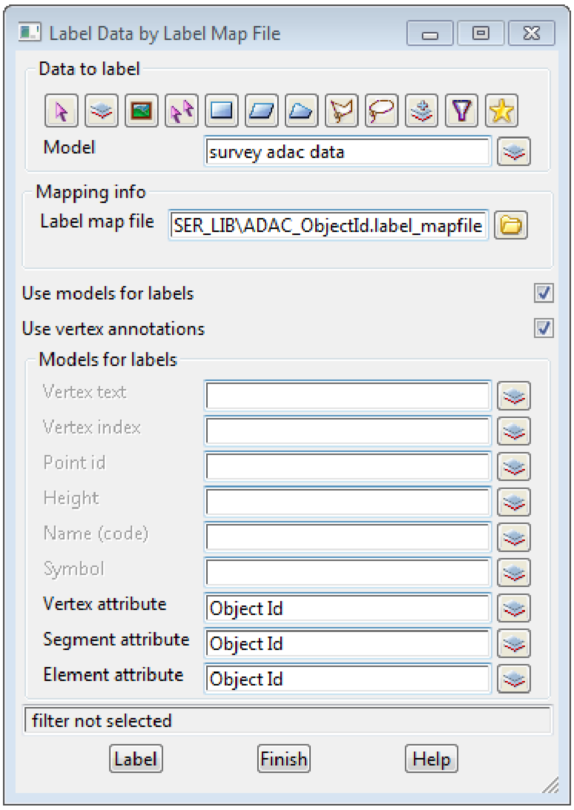
Symbol:

Vertex attribute: Object Id

Segment attribute: Object Id

Element attribute: Object Id

filter not selected



Label Data by Label Map File

Data to label

Model: survey adac data

Mapping info

Label map file: SER_LIB\ADAC_ObjectId.label_mapfile

Use models for labels ☑

Use vertex annotations ☑

Models for labels

Vertex text:

Vertex index:

Point id:

Height:

Name (code):

Symbol:

Vertex attribute: Object Id

Segment attribute: Object Id

Element attribute: Object Id

filter not selected

Click on **Label** and when the labelling is finished, click on **Finish**.

The chain then ends.

Continue to the next section 1.4.9.3.4 Show Attributed/Not Attributed or return to 1.4.9.3 User ADAC Utilities .

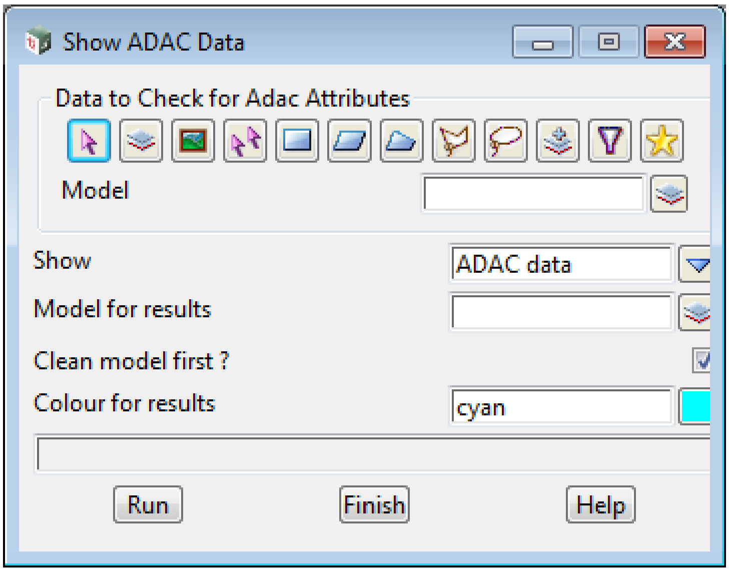
1.4.9.3.4 Show Attributed/Not Attributed

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



Show: ADAC data

Model for results:

Clean model first ? ☑

Colour for results: cyan

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 check if it has/doesn't have ADAC attributes.

Show choice box ADAC data ADAC data, non ADAC data

*if **ADAC data**, 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* **Colour for results** *and is added to the model in* **Model for results.**

*if **non ADAC data**, 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* **Colour for results** *and is added to the model in* **Model for results.**

Model for results model box available models

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.*

Colour for results colour box cyan available colours

the colour to give the created strings.

Run button

finds all the ADAC/not ADAC strings.

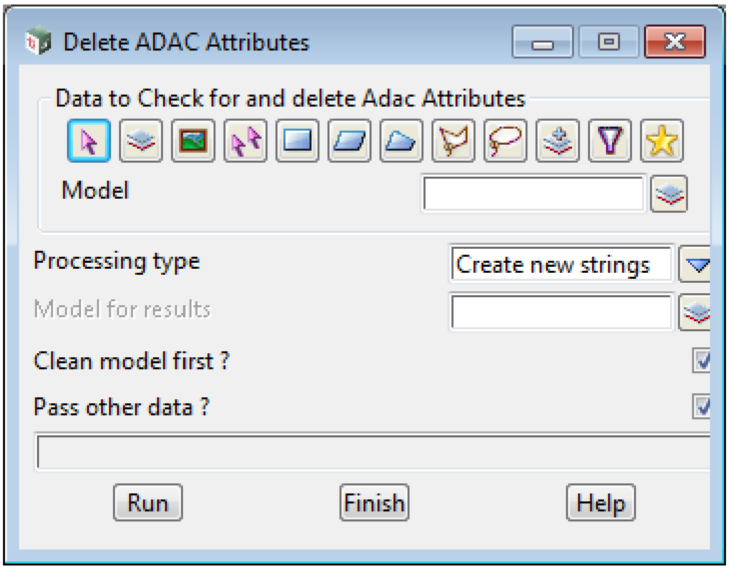
Continue to 1.4.9.3.5 Delete ADAC Attributes or return to 1.4.9.3 User ADAC Utilities .

1.4.9.3.5 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:



Delete ADAC Attributes

Data to Check for and delete Adac Attributes

Model

Processing type: Create new strings

Model for results

Clean model first ? ☑

Pass other data ? ☑

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.

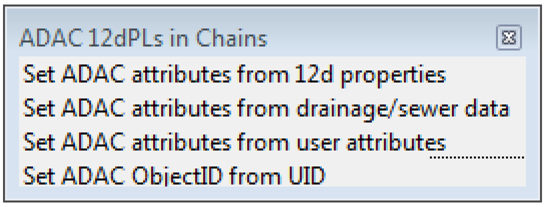
Run button

removed the ADAC attributes from the selected strings.

Continue to 1.4.9.3.6 12dPL's in Chains or return to 1.4.9.3 User ADAC Utilities .

1.4.9.3.6 12dPL's in Chains

Position of menu: **File I/O =>ADAC =>User =>Utilities**


ADAC 12dPLs in Chains
Set ADAC attributes from 12d properties
Set ADAC attributes from drainage/sewer data
Set ADAC attributes from user attributes
Set ADAC ObjectID from UID

There are the options used in the ADAC Design and Survey chains.

See

1.4.9.3.6.1 Separate Assigned/Unassigned ADAC Data

1.4.9.3.6.2 Set ADAC Attributes from String Properties

1.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data

1.4.9.3.6.4 Set ADAC Attributes from User Attributes

1.4.9.3.6.5 Set ADAC ObjectId from String UID

1.4.9.3.6.5 Set ADAC ObjectId from String UID

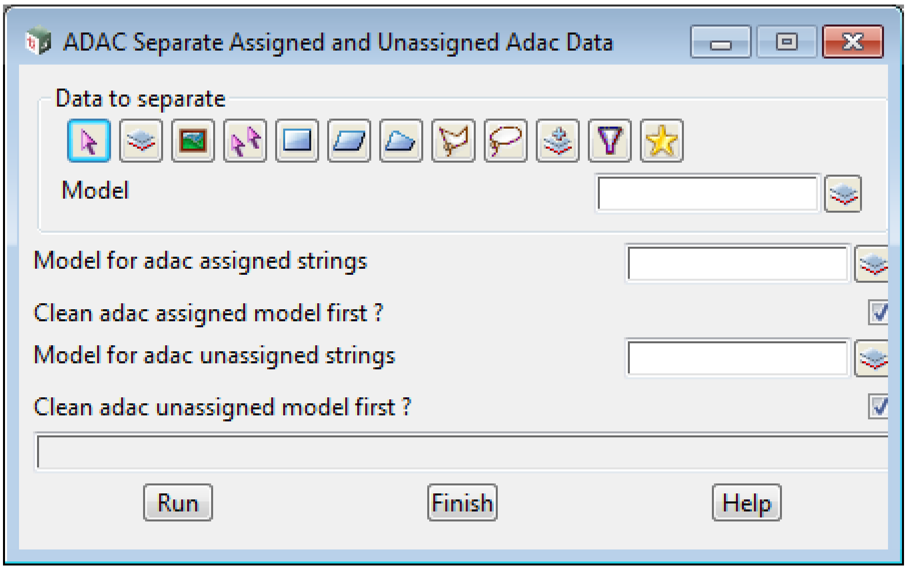
1.4.9.3.6.1 Separate Assigned/Unassigned ADAC Data

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.



ADAC Separate Assigned and Unassigned Adac Data

Data to separate

Model

Model for adac assigned strings

Clean adac assigned model first ? ☑

Model for adac unassigned strings

Clean adac unassigned model first ? ☑

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 1.4.9.3.6.2 Set ADAC Attributes from String Properties or return to 1.4.9.3.6 12dPL's in Chains

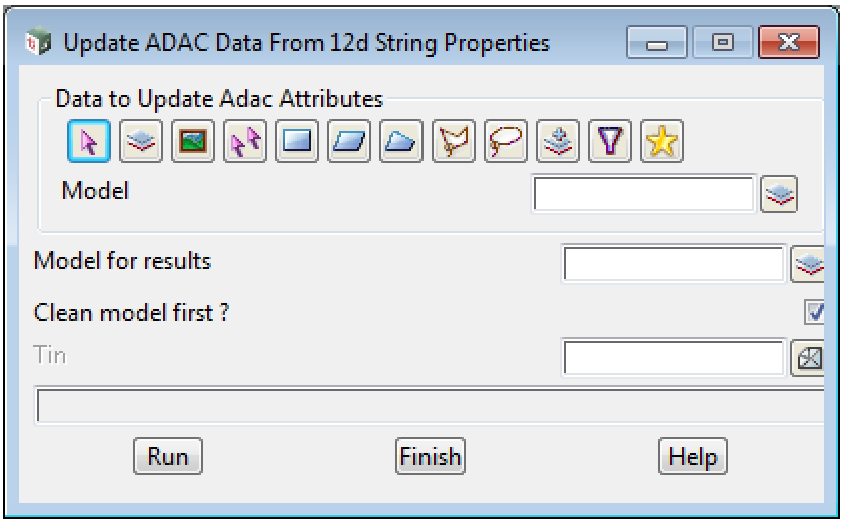
1.4.9.3.6.2 Set ADAC Attributes from String Properties

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.


Update ADAC Data From 12d String Properties
Data to Update Adac Attributes
Model
Model for results
Clean model first ?
Tin
Run
Finish
Help

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.

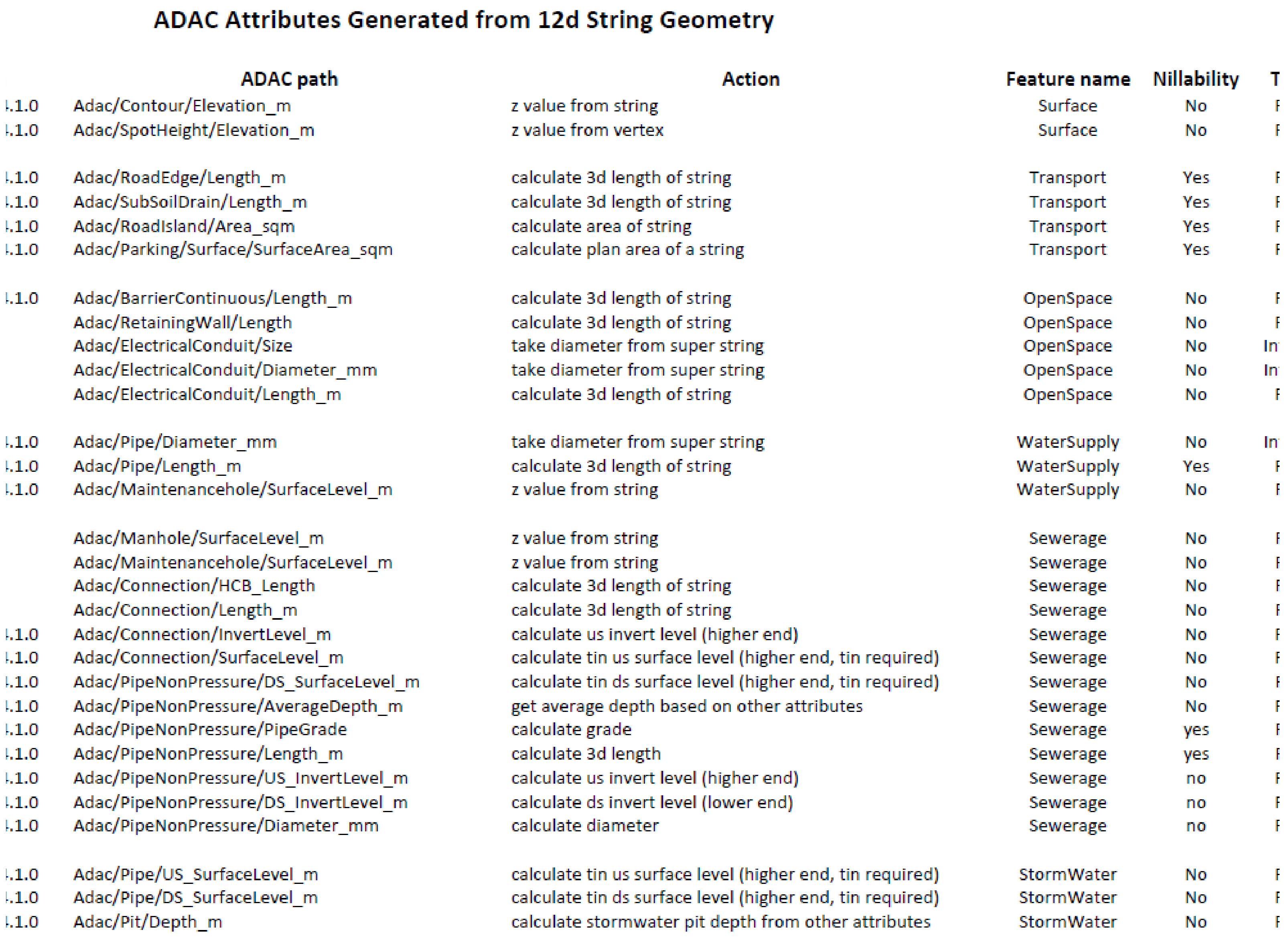
Run button

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.

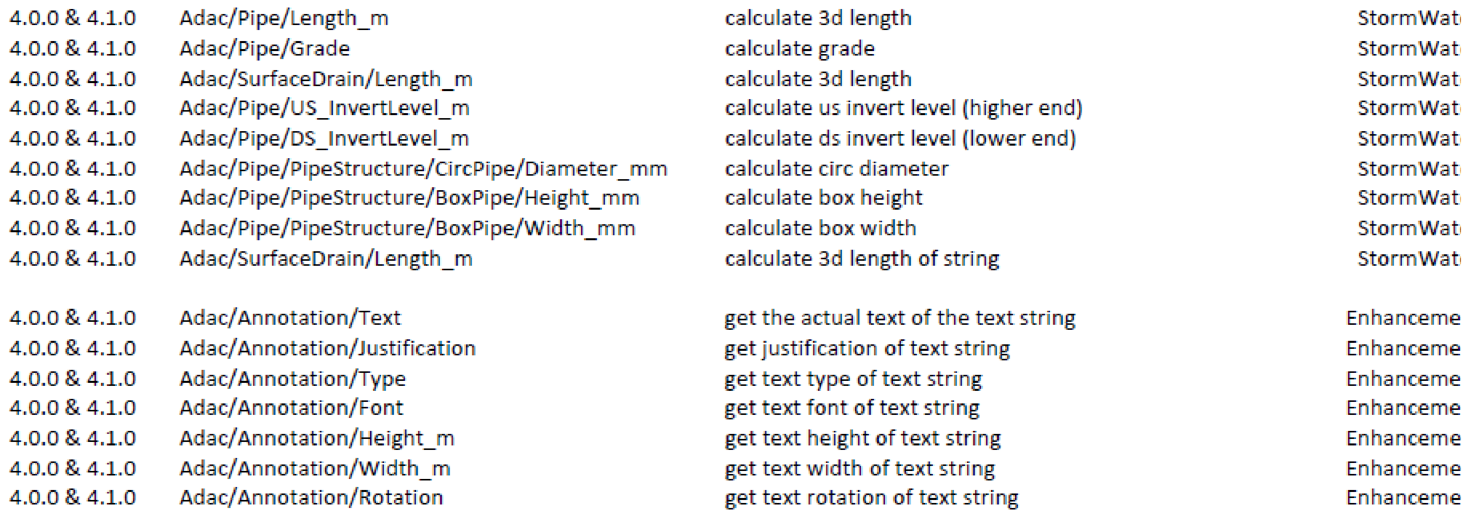
Strings that are not marked as ADAC Assets are NOT copied

The copied strings are added to the model ***Model for results****.*



ADAC Attributes Generated from 12d String Geometry

	ADAC path	Action	Feature name	Nillability	T
.1.0	Adac/Contour/Elevation_m	z value from string	Surface	No	F
.1.0	Adac/SpotHeight/Elevation_m	z value from vertex	Surface	No	F
.1.0	Adac/RoadEdge/Length_m	calculate 3d length of string	Transport	Yes	F
.1.0	Adac/SubSoilDrain/Length_m	calculate 3d length of string	Transport	Yes	F
.1.0	Adac/RoadIsland/Area_sqm	calculate area of string	Transport	Yes	F
.1.0	Adac/Parking/Surface/SurfaceArea_sqm	calculate plan area of a string	Transport	Yes	F
.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
.1.0	Adac/Pipe/Diameter_mm	take diameter from super string	WaterSupply	No	In
.1.0	Adac/Pipe/Length_m	calculate 3d length of string	WaterSupply	Yes	F
.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
.1.0	Adac/Connection/InvertLevel_m	calculate us invert level (higher end)	Sewerage	No	F
.1.0	Adac/Connection/SurfaceLevel_m	calculate tin us surface level (higher end, tin required)	Sewerage	No	F
.1.0	Adac/PipeNonPressure/DS_SurfaceLevel_m	calculate tin ds surface level (higher end, tin required)	Sewerage	No	F
.1.0	Adac/PipeNonPressure/AverageDepth_m	get average depth based on other attributes	Sewerage	No	F
.1.0	Adac/PipeNonPressure/PipeGrade	calculate grade	Sewerage	yes	F
.1.0	Adac/PipeNonPressure/Length_m	calculate 3d length	Sewerage	yes	F
.1.0	Adac/PipeNonPressure/US_InvertLevel_m	calculate us invert level (higher end)	Sewerage	no	F
.1.0	Adac/PipeNonPressure/DS_InvertLevel_m	calculate ds invert level (lower end)	Sewerage	no	F
.1.0	Adac/PipeNonPressure/Diameter_mm	calculate diameter	Sewerage	no	F
.1.0	Adac/Pipe/US_SurfaceLevel_m	calculate tin us surface level (higher end, tin required)	StormWater	No	F
.1.0	Adac/Pipe/DS_SurfaceLevel_m	calculate tin ds surface level (higher end, tin required)	StormWater	No	F
.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/PipeStructure/CircPipe/Diameter_mm	calculate circ diameter	StormWat
4.0.0 & 4.1.0	Adac/Pipe/PipeStructure/BoxPipe/Height_mm	calculate box height	StormWat
4.0.0 & 4.1.0	Adac/Pipe/PipeStructure/BoxPipe/Width_mm	calculate box width	StormWat
4.0.0 & 4.1.0	Adac/SurfaceDrain/Length_m	calculate 3d length of string	StormWat
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 1.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey *and* 1.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design .

Continue to the next section 1.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data or return to 1.4.9.3.6 12dPL's in Chains .

1.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data

Position of option on menu:

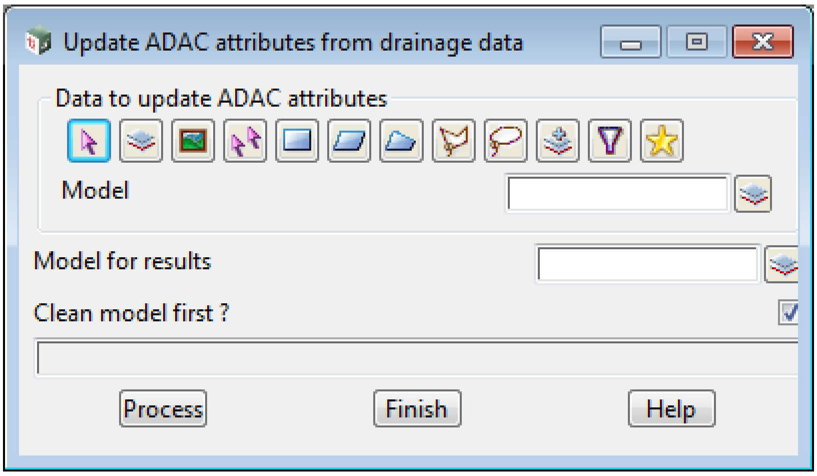
File I/O =>ADAC =>User =>Utilities =>12dPLs in chains =>Set ADAC attributes from drainage/ sewer data

The documentation for this section is a work in progress.

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:



Update ADAC attributes from drainage data

Data to update ADAC attributes

Model

Model for results

Clean model first ? ☑

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 1.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties .

Continue to the next section 1.4.9.3.6.5 Set ADAC ObjectId from String UID or return to 1.4.9.3.6 12dPL's in Chains .

1.4.9.3.6.4 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

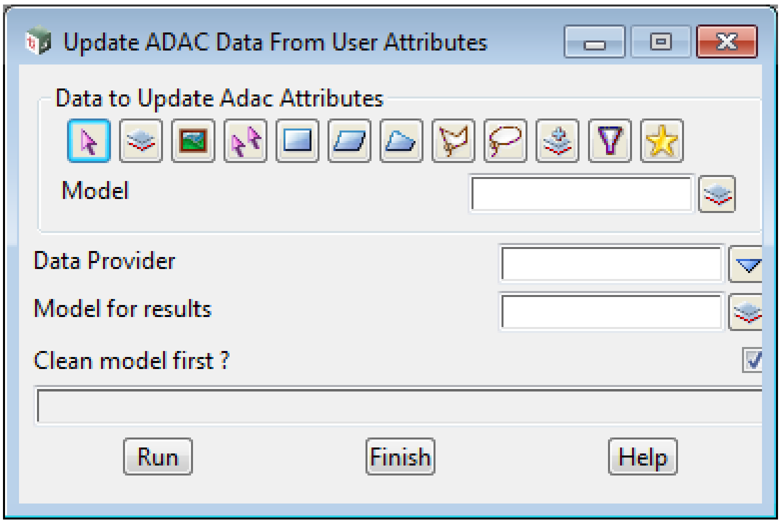
The option is no longer used and had been replaced by a 12d supplied **12duaf file**. See 1.4.8.5 Create/Edit User Attributes to ADAC File and 1.4.8.6 Apply User Attributes to ADAC Elements .

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.


Update ADAC Data From User Attributes
Data to Update Adac Attributes
Model
Data Provider
Model for results
Clean model first ?
Run
Finish
Help

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 1.4.9.3.6.5 Set ADAC ObjectId from String UID or return to 1.4.9.3.6 12dPL's in Chains .

1.4.9.3.6.5 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

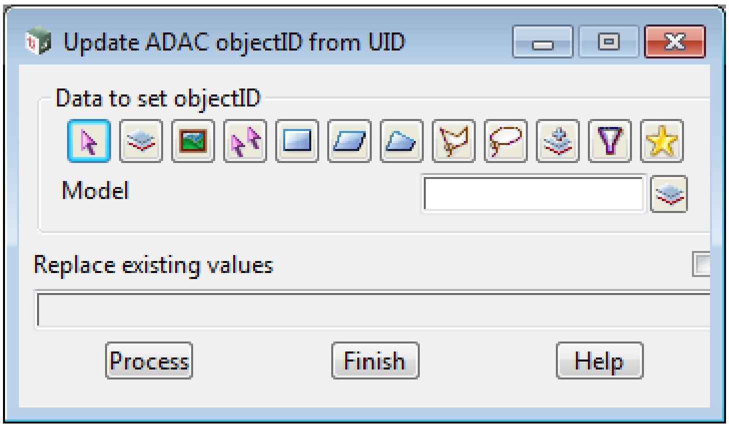
(i) each pit/manhole. The ObjectId is made up of the string UID, the pit/manhole vertex index and information about the to help connect the string together.

(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 1.5.1.2.6 Update ADAC ObjectId - Survey *and* 1.5.2.2.7 Update ADAC ObjectId - Design .

Return to 1.4.9.3.6 12dPL's in Chains .

1.4.9.4 12d 4.1 Chains

Position of menu: **File I/O =>ADAC =>User =>12d 4.1 chains**

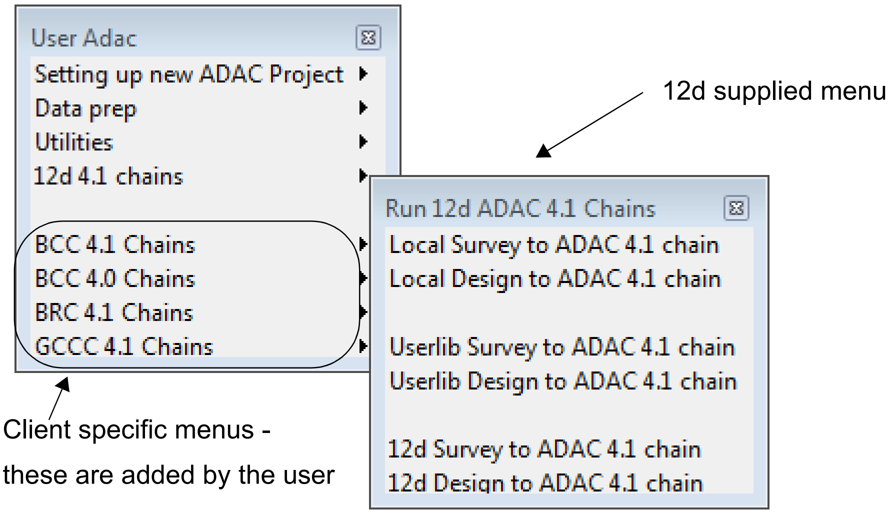
There is a base *ADAC Design* chain and a base *ADAC Survey* chain that does all the work of taking ***12d Model*** strings and producing ADAC Assets ready to be validated, reported on, or written out to an ADAC XML file.

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.


User Adac
Setting up new ADAC Project
Data prep
Utilities
12d 4.1 chains
BCC 4.1 Chains
BCC 4.0 Chains
BRC 4.1 Chains
GCCC 4.1 Chains
Run 12d ADAC 4.1 Chains
Local Survey to ADAC 4.1 chain
Local Design to ADAC 4.1 chain
Userlib Survey to ADAC 4.1 chain
Userlib Design to ADAC 4.1 chain
12d Survey to ADAC 4.1 chain
12d Design to ADAC 4.1 chain

12d supplied menu

Client specific menus -

these are added by the user

See

1.4.9.4.1 Local Survey to ADAC 41 chain
1.4.9.4.2 Local Design to ADAC 41 chain

1.4.9.4.3 Userlib Survey to ADAC 41 chain
1.4.9.4.4 Userlib Design to ADAC 41 chain

1.4.9.4.5 Lib Survey to ADAC 41 chain
1.4.9.4.6 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 1.4.9.5 Client Specific ADAC Design & Survey Menus .

1.4.9.4.1 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 1.4.9.4.2 Local Design to ADAC 41 chain or return to 1.4.9.4 12d 4.1 Chains .

1.4.9.4.2 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 1.4.9.4.3 Userlib Survey to ADAC 41 chain or return to 1.4.9.4 12d 4.1 Chains .

1.4.9.4.3 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 1.4.9.4.4 Userlib Design to ADAC 41 chain or return to 1.4.9.4 12d 4.1 Chains .

1.4.9.4.4 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 1.4.9.4.5 Lib Survey to ADAC 41 chain or return to 1.4.9.4 12d 4.1 Chains .

1.4.9.4.5 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 1.4.9.4.6 Lib Design to ADAC 41 chain or return to 1.4.9.4 12d 4.1 Chains .

1.4.9.4.6 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 1.4.9.4 12d 4.1 Chains .

1.4.9.5 Client Specific ADAC Design & Survey Menus

Position of menu: **File I/O =>ADAC =>User =>Client chains**

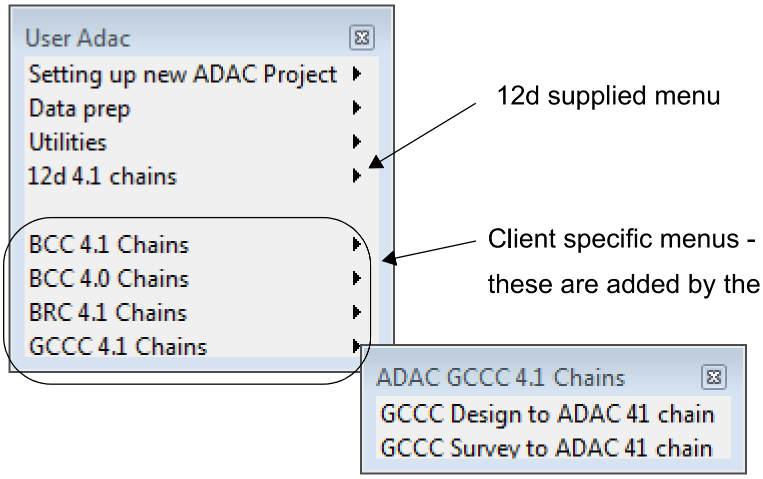
There is a base ADAC Design chain and a base ADAC Survey chain that does all the work of taking ***12d Model*** strings and producing ADAC Assets ready to be validated, reported on, or written out to an ADAC XML file.

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:



12d supplied menu

Client specific menus -

these are added by the user

How to set up these user defined ADAC menus for your company is described in 1.6.3.2 Setting Up Your User ADAC Menu .

Note: **12d Supplied ADAC 4.1 Design and Survey Menu**

If you are 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 1.4.9.4 12d 4.1 Chains .

Return to 1.4.9.3 User ADAC Utilities .

1.5 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 1.3 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 1.4.2 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

1.5.1 Survey to ADAC Chains

1.5.2 Design to ADAC Chains

1.5.1 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

1.5.1.1 ADAC Survey Base Chain pvf File

1.5.1.2 ADAC Survey Base Chain

1.5.1.1 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.

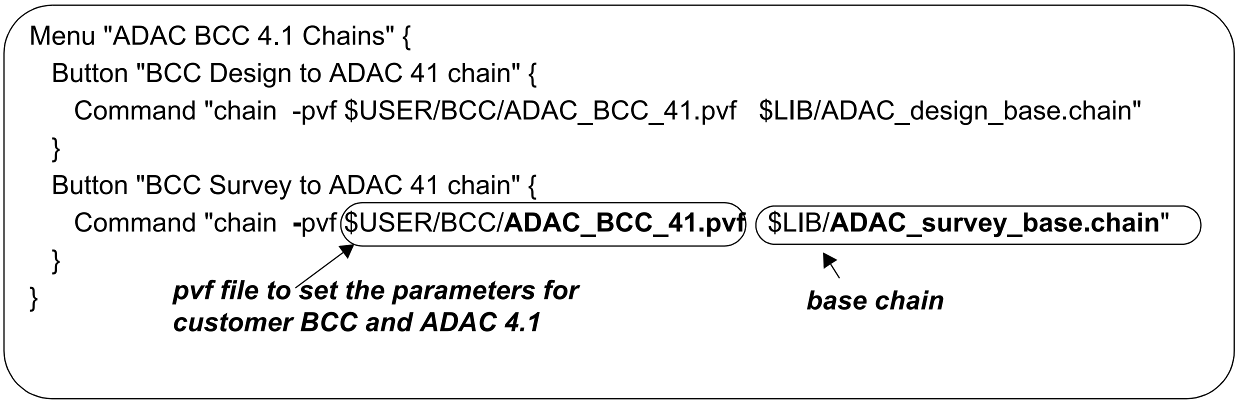
There are two parameters to define which ***12d User Adac to ADAC Elements Files*** to use and again they are fully determined once *company* and *adac_version_by_ten* have values.

8. The text parameter **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"
  }
}
```

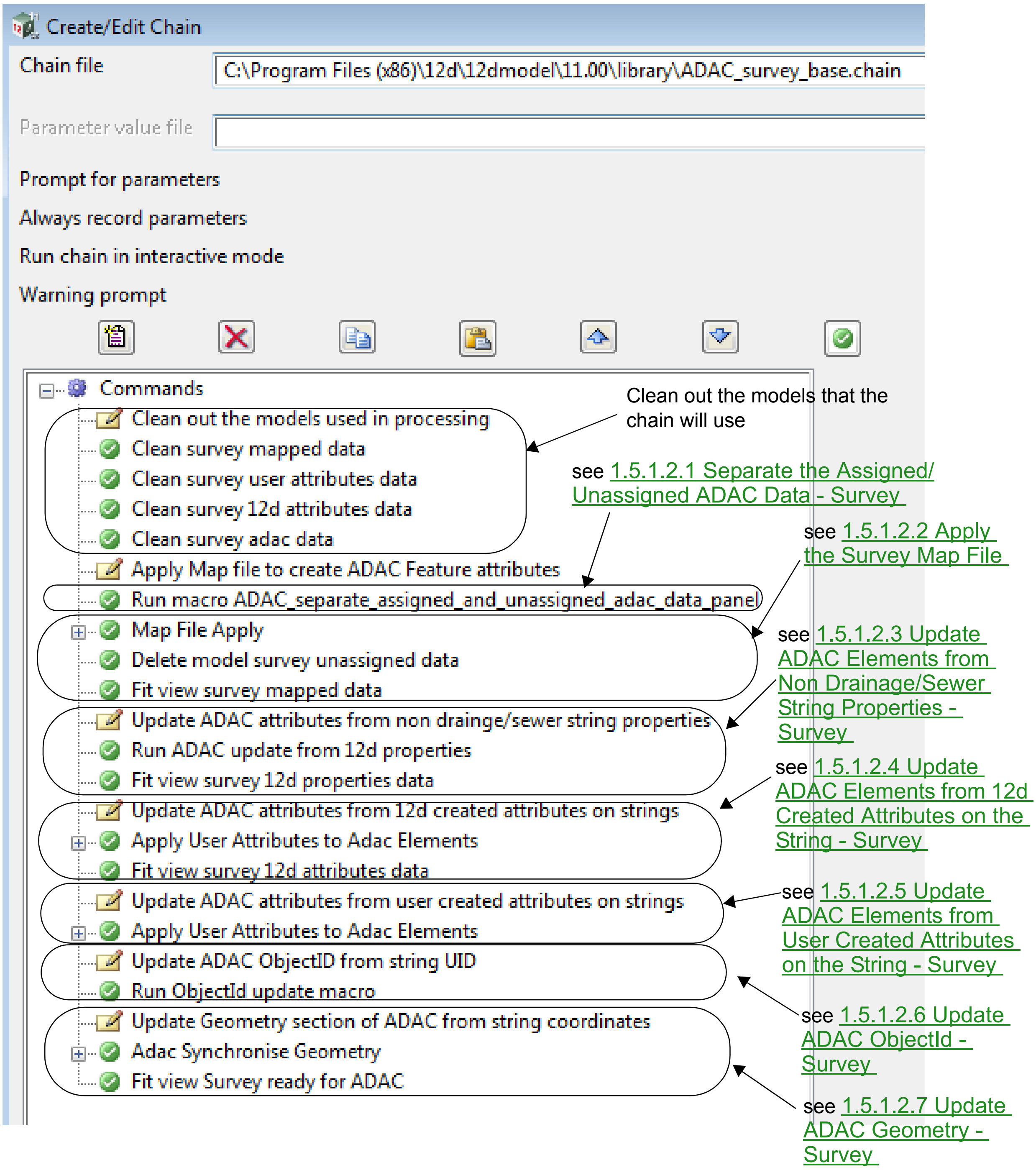
pvf file to set the parameters for customer BCC and ADAC 4.1

base chain

ADAC_Survey_base.chain is discussed in detail in 1.5.1.2 ADAC Survey Base Chain

1.5.1.2 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**.



Create/Edit Chain

Chain file C:\Program Files (x86)\12d\12dmodel\11.00\library\ADAC_survey_base.chain

Parameter value file

Prompt for parameters

Always record parameters

Run chain in interactive mode

Warning prompt

Commands
- 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 drainge/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

Clean out the models that the chain will use

see 1.5.1.2.1 Separate the Assigned/Unassigned ADAC Data - Survey

see 1.5.1.2.2 Apply the Survey Map File

see 1.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey

see 1.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey

see 1.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey

see 1.5.1.2.6 Update ADAC ObjectId - Survey

see 1.5.1.2.7 Update ADAC Geometry - Survey

1.5.1.2.1 Separate the Assigned/Unassigned ADAC Data - Survey

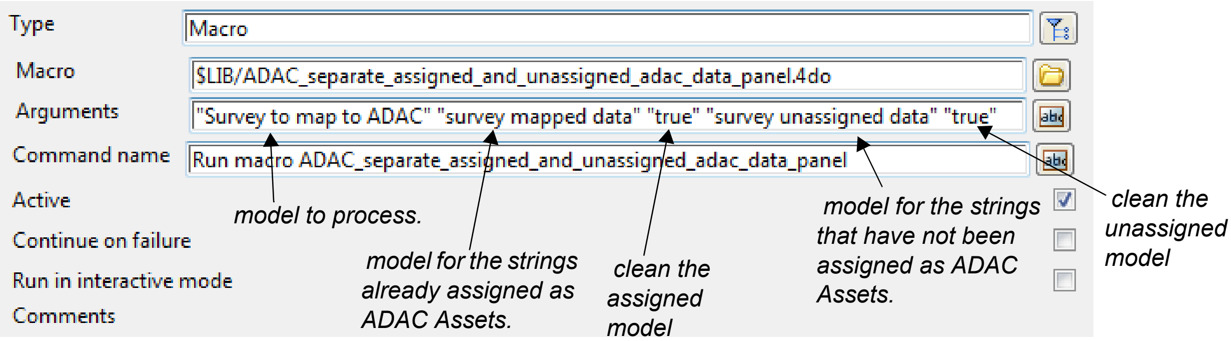
Some strings on the **View** called **Survey to map to ADAC** may have already been assigned (marked) as ADAC Assets and so should not have the ADAC Map file applied to them.

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**.



Type: Macro

Macro: $LIB/ADAC_separate_assigned_and_unassigned_adac_data_panel.4do

Arguments: "Survey to map to ADAC" "survey mapped data" "true" "survey unassigned data" "true"

Command name: Run macro ADAC_separate_assigned_and_unassigned_adac_data_panel

Active ☑

Continue on failure □

Run in interactive mode □

Comments

model to process.

model for the strings already assigned as ADAC Assets.

clean the assigned model

model for the strings that have not been assigned as ADAC Assets.

clean the unassigned model

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 1.4.9.3.6.1 Separate Assigned/ Unassigned ADAC Data

Continue to the next section 1.5.1.2.2 Apply the Survey Map File or return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

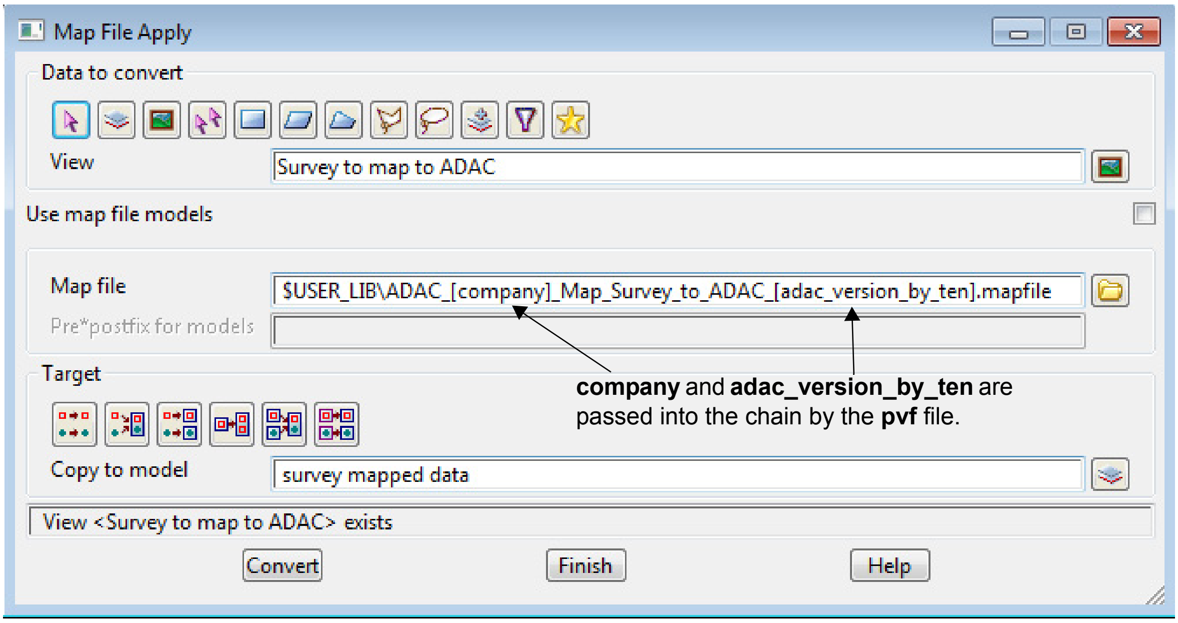
1.5.1.2.2 Apply the Survey Map File

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**.



View: Survey to map to ADAC

Use map file models: □

Map file: $USER_LIB\ADAC_[company]_Map_Survey_to_ADAC_[adac_version_by_ten].mapfile

Pre*postfix for models:

Copy to model: survey mapped data

View <Survey to map to ADAC> exists

company and **adac_version_by_ten** are passed into the chain by the **pvf** 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 1.5.1.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Survey or return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

1.5.1.2.3 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.

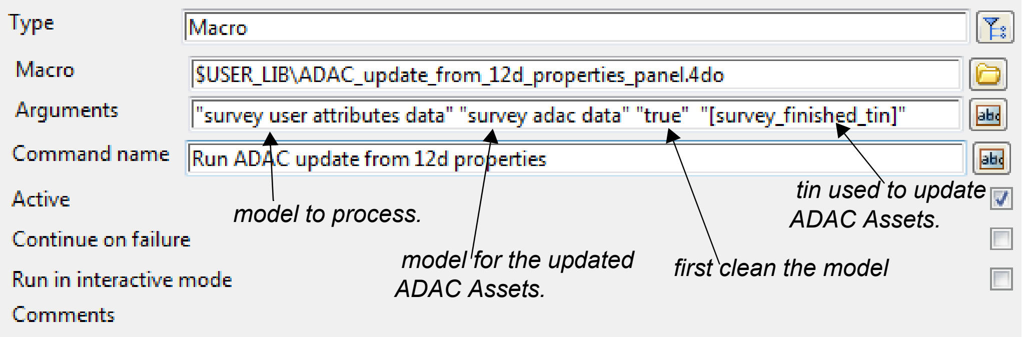
This is achieved by running the 12dPL (macro)

$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 the12d 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.**



Type	Macro
Macro	$USER_LIB\ADAC_update_from_12d_properties_panel.4do
Arguments	"survey user attributes data" "survey adac data" "true" "[survey_finished_tin]"
Command name	Run ADAC update from 12d properties
Active	☑
Continue on failure	□
Run in interactive mode	□
Comments	

model to process.

model for the updated ADAC Assets.

first clean the model

tin used to update ADAC Assets.

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 1.4.9.3.6.2 Set ADAC Attributes from String Properties

Continue to the next section 1.5.1.2.4 Update ADAC Elements from 12d Created Attributes on the String - Survey or return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

1.5.1.2.4 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 1.4.9.2.1 Providing Extra Data for ADAC .

This is achieved by running the panel

ADAC User Attributes to ADAC Element

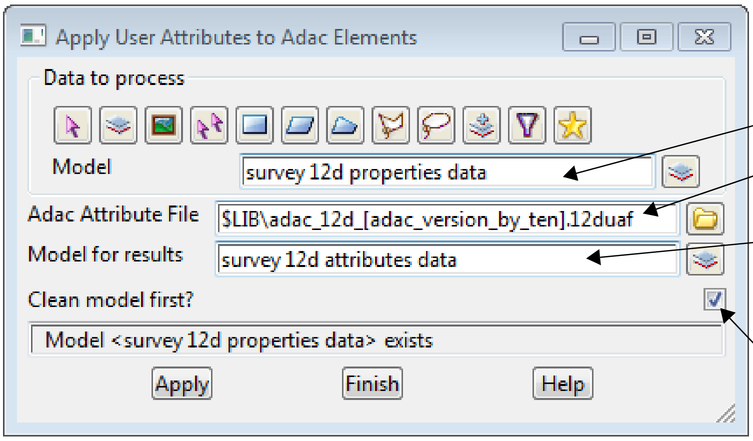
with a *12d Solutions* supplied *12duaf* file for the required version of ADAC.

(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**.



Apply User Attributes to Adac Elements

Data to process

Model: survey 12d properties data

Adac Attribute File: $LIB\adac_12d_[adac_version_by_ten].12duaf

Model for results: survey 12d attributes data

Clean model first? ☑

Model <survey 12d properties data> exists

model to process.

12duaf file to use - supplied by 12d Solutions

model for the updated ADAC Assets.

first clean the model

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 1.4.8.6 Apply User Attributes to ADAC Elements

Continue to the next section 1.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey or return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

1.5.1.2.5 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 updated 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

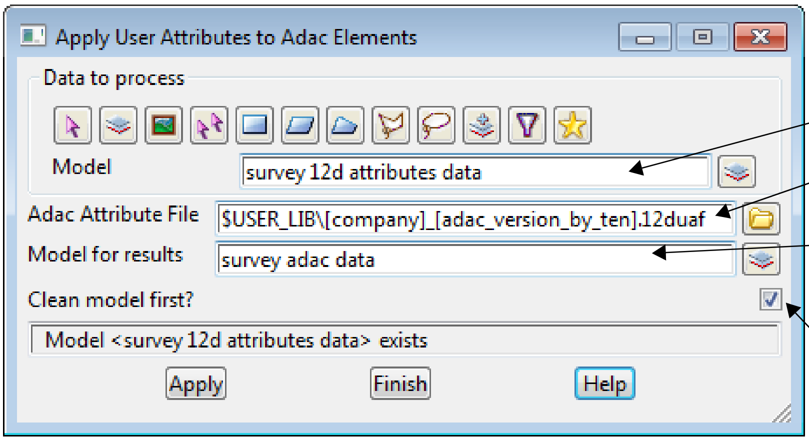
ADAC User Attributes to ADAC Element

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**.



Apply User Attributes to Adac Elements

Data to process

Model survey 12d attributes data

Adac Attribute File $USER_LIB\[company]_[adac_version_by_ten].12duaf

Model for results survey adac data

Clean model first? ☑

Model <survey 12d attributes data> exists

Apply
Finish
Help
model to process.
company supplied 12duaf file to use
model for the updated ADAC Assets.
first clean the model

The model **survey user attributes data** is on the view called **survey string properties data.**

Note - this option is described in more detail in 1.4.8.6 Apply User Attributes to ADAC Elements

Continue to the next section 1.5.1.2.6 Update ADAC ObjectId - Survey or return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

1.5.1.2.6 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.

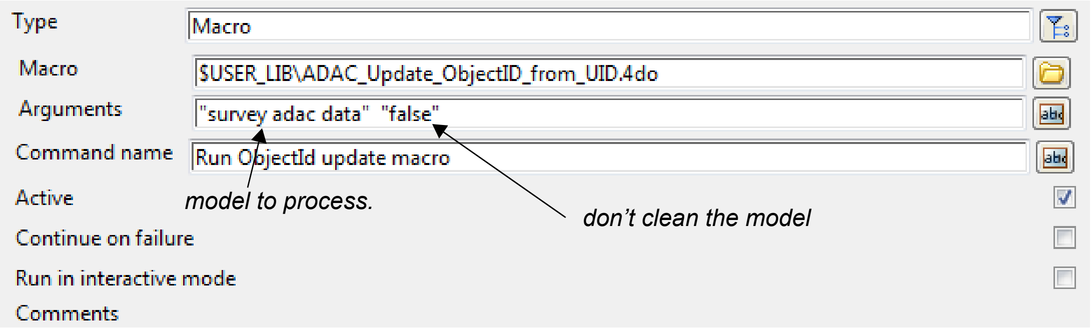
This is only relevant for designers and not surveyors.

(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_panel.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.



Type Macro
Macro $USER_LIB\ADAC_Update_ObjectID_from_UID.4do
Arguments "survey adac data" "false"
Command name Run ObjectId update macro
Active ☑
Continue on failure ☐
Run in interactive mode ☐
Comments

model to process.

don't clean the model

The model **survey adac data** is on the view called **survey ready for adac.**

Note - this *12dPL* program is described in more detail in 1.4.9.3.6.5 Set ADAC ObjectId from String UID

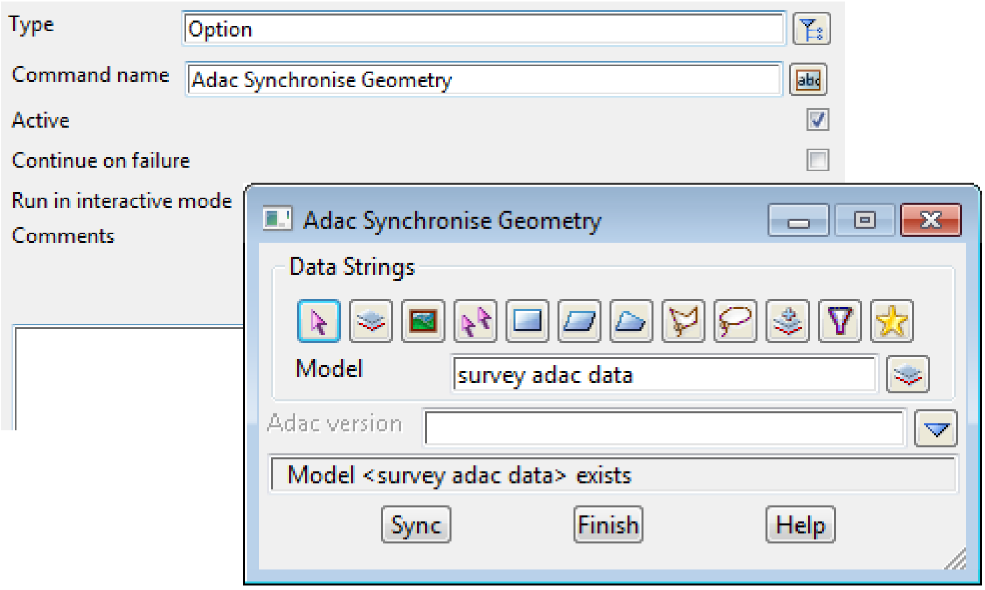
Continue to the next section 1.5.1.2.7 Update ADAC Geometry - Survey or return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

1.5.1.2.7 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



Type: Option

Command name: Adac Synchronise Geometry

Active ☑

Continue on failure □

Run in interactive mode

Comments

Adac Synchronise Geometry

Data Strings

Model: survey adac data

Adac version:

Model <survey adac data> exists

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 1.4.8.4 Sync Geometry

Return to 1.5.1.2 ADAC Survey Base Chain or 1.5.1 Survey to ADAC Chains .

1.5.2 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

1.5.2.1 ADAC Design Base Chain pvf File

1.5.2.2 ADAC Design Base Chain

1.5.2.1 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.

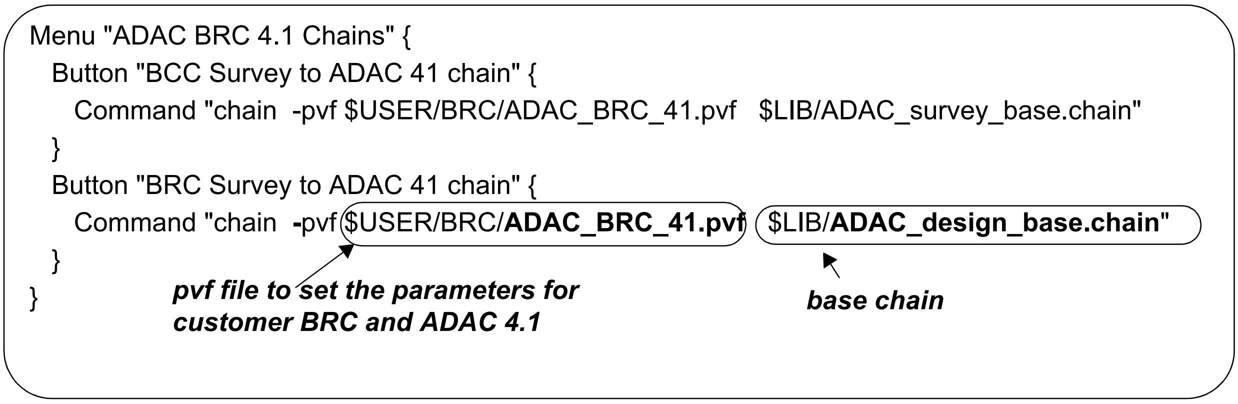
8. The text parameter **adac_attribute_file** has the value $USER_LIB\[company]_[adac_version_by_ten].12duaf

   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.



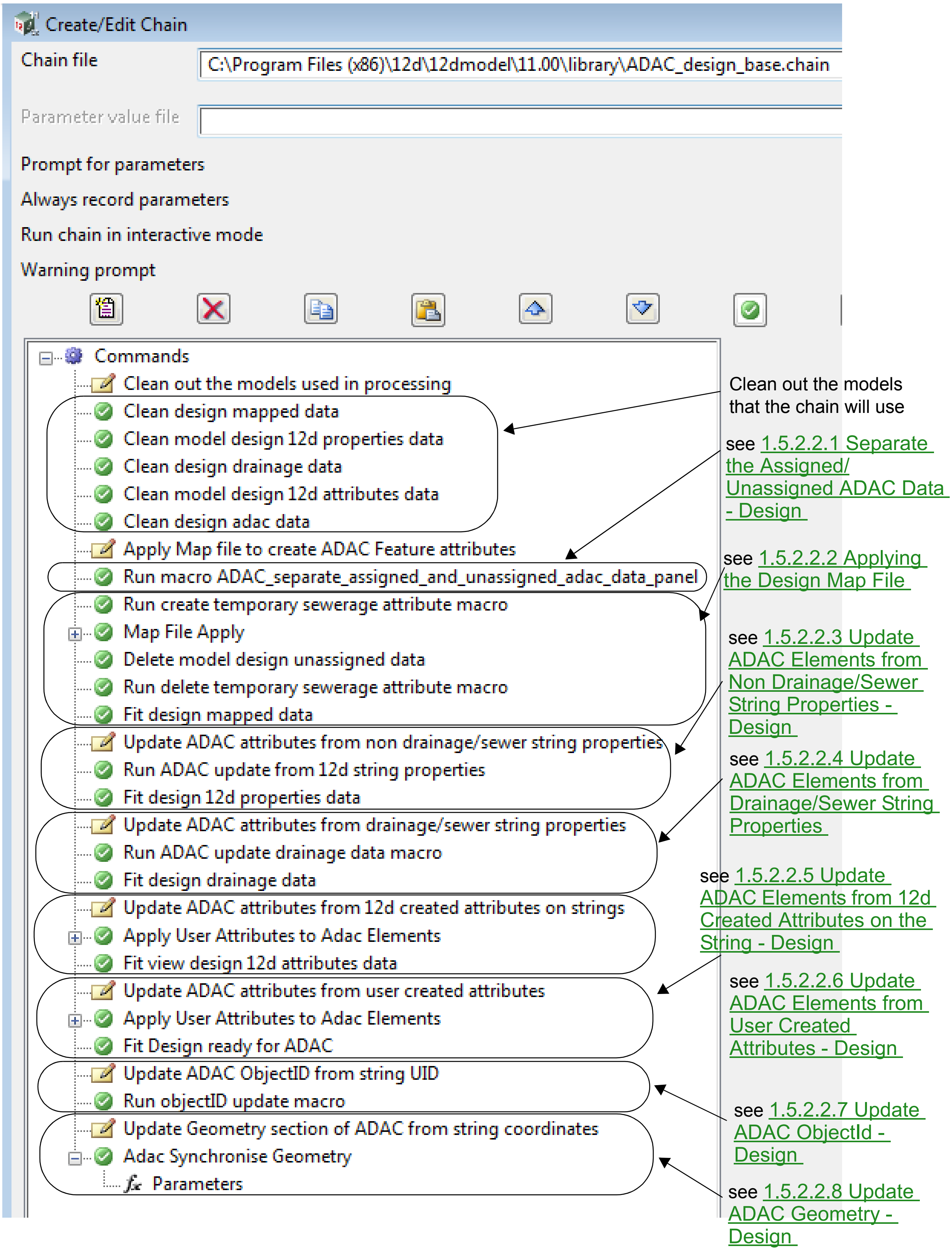
```
Menu "ADAC BRC 4.1 Chains" {
  Button "BCC Survey to ADAC 41 chain" {
    Command "chain  -pvf $USER/BRC/ADAC_BRC_41.pvf   $LIB/ADAC_survey_base.chain"
  }
  Button "BRC Survey to ADAC 41 chain" {
    Command "chain  -pvf $USER/BRC/ADAC_BRC_41.pvf  $LIB/ADAC_design_base.chain"
  }
}
```

pvf file to set the parameters for customer BRC and ADAC 4.1
base chain

ADAC_Design_base.chain is discussed in detail in 1.5.2.2 ADAC Design Base Chain

1.5.2.2 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

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
```

Clean out the models that the chain will use

see 1.5.2.2.1 Separate the Assigned/ Unassigned ADAC Data - Design

see 1.5.2.2.2 Applying the Design Map File

see 1.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design

see 1.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties

see 1.5.2.2.5 Update ADAC Elements from 12d Created Attributes on the String - Design

see 1.5.2.2.6 Update ADAC Elements from User Created Attributes - Design

see 1.5.2.2.7 Update ADAC ObjectId - Design

see 1.5.2.2.8 Update ADAC Geometry - Design

1.5.2.2.1 Separate the Assigned/Unassigned ADAC Data - Design

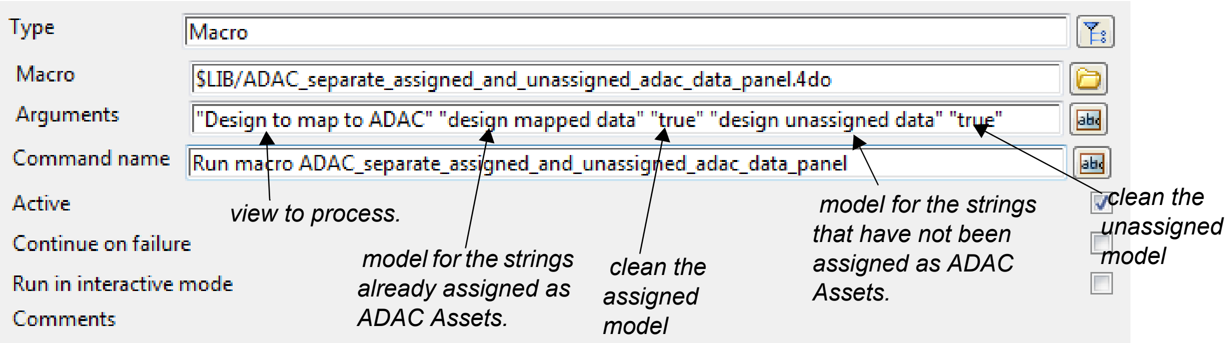
Some strings on the **View** called **Survey to map to ADAC** may have already been assigned (marked) as ADAC Assets and so should not have the ADAC Map file applied to them.

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**.



Type	Macro
Macro	$LIB/ADAC_separate_assigned_and_unassigned_adac_data_panel.4do
Arguments	"Design to map to ADAC" "design mapped data" "true" "design unassigned data" "true"
Command name	Run macro ADAC_separate_assigned_and_unassigned_adac_data_panel
Active	☑
Continue on failure	□
Run in interactive mode	□
Comments	

view to process.

model for the strings already assigned as ADAC Assets.

clean the assigned model

model for the strings that have not been assigned as ADAC Assets.

clean the unassigned model

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 1.4.9.3.6.1 Separate Assigned/ Unassigned ADAC Data

Continue to the next section 1.5.2.2.2 Applying the Design Map File or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.2 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

(a) **drainage string**, creates a **string attribute** called **sewertype** of type Integer with 0.

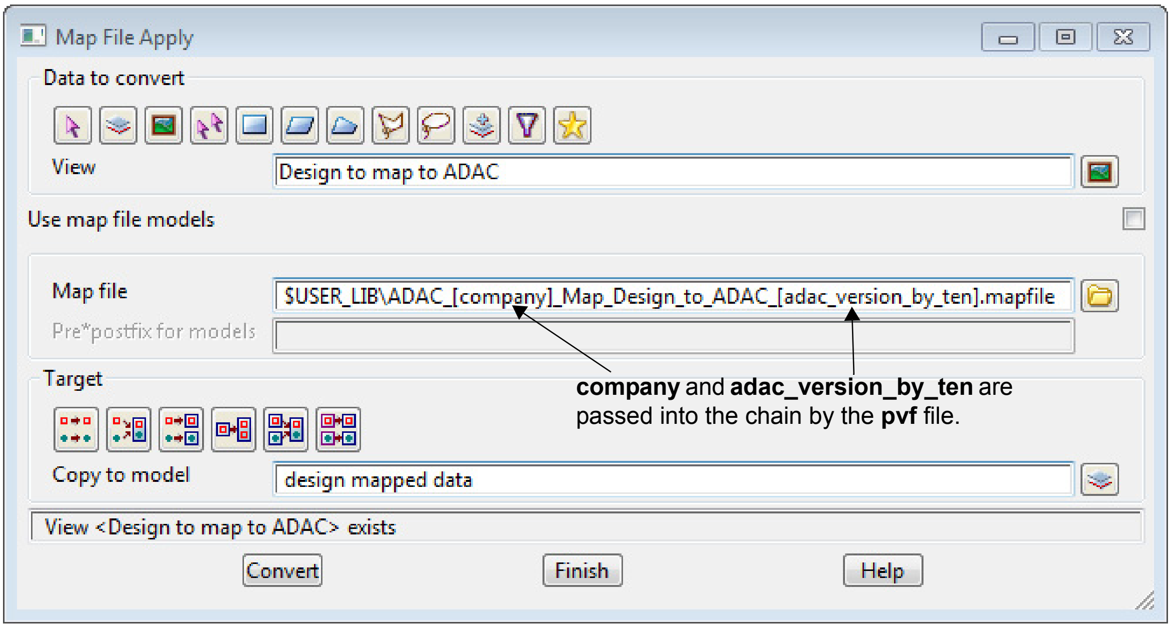
(b) **sewer string**, create a **string attribute** called **sewertype** of type Integer with value 0.

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***.


Map File Apply

Data to convert

View: Design to map to ADAC

Use map file models □

Map file: $USER_LIB\ADAC_[company]_Map_Design_to_ADAC_[adac_version_by_ten].mapfile

Pre*postfix for models:

Target

Copy to model: design mapped data

View <Design to map to ADAC> exists

company and **adac_version_by_ten** are passed into the chain by the **pvf** 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 1.5.2.2.3 Update ADAC Elements from Non Drainage/Sewer String Properties - Design or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.3 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.

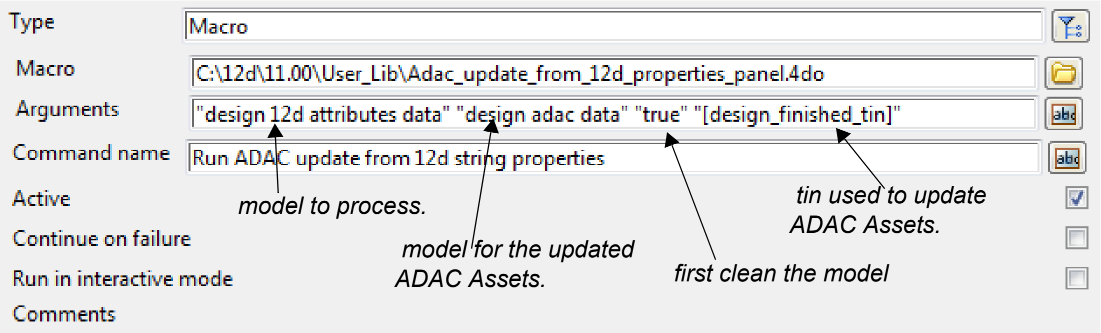
This is achieved by running the 12dPL (macro)

$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 the12d 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.**



Type: Macro

Macro: C:\12d\11.00\User_Lib\Adac_update_from_12d_properties_panel.4do

Arguments: "design 12d attributes data" "design adac data" "true" "[design_finished_tin]"

Command name: Run ADAC update from 12d string properties

Active ☑

Continue on failure □

Run in interactive mode □

Comments

model to process.

model for the updated ADAC Assets.

first clean the model

tin used to update ADAC Assets.

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 1.4.9.3.6.2 Set ADAC Attributes from String Properties

Continue to the next section 1.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.4 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.

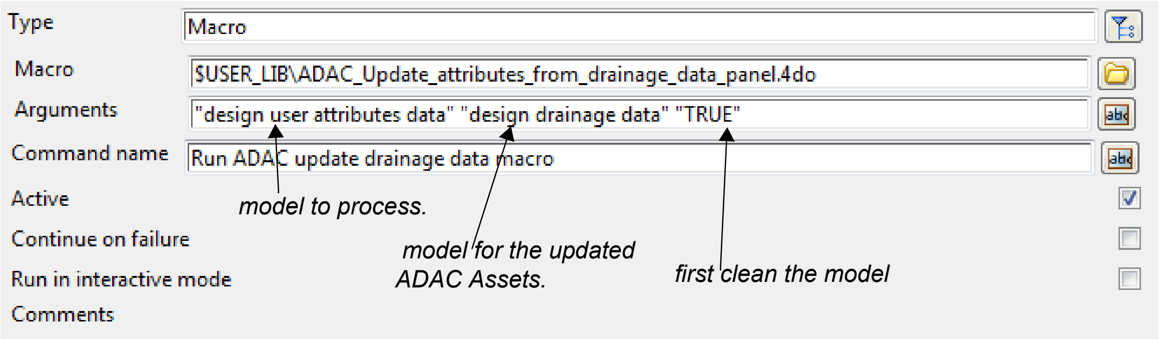
This is achieved by running the 12dPL (macro)

$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 the12d string itself.

The copied string is added to the model **design drainage data**.

Only ADAC Assets are added to the model **design drainage data.**



Type Macro

Macro $USER_LIB\ADAC_Update_attributes_from_drainage_data_panel.4do

Arguments "design user attributes data" "design drainage data" "TRUE"

Command name Run ADAC update drainage data macro

Active ☑

Continue on failure □

Run in interactive mode □

Comments

model to process.

model for the updated ADAC Assets.

first clean the model

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 1.4.9.3.6.3 Set ADAC Attributes from Drainage & Sewer Data

Continue to the next section 1.5.2.2.5 Update ADAC Elements from 12d Created Attributes on the String - Design or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.5 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 1.4.9.2.1 Providing Extra Data for ADAC .

This is achieved by running the panel

ADAC User Attributes to ADAC Element

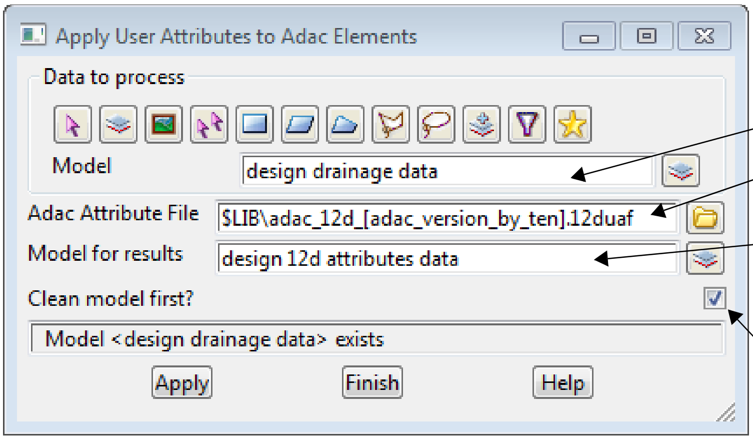
with a *12d Solutions* supplied *12duaf* file for the required version of ADAC.

(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**.


Apply User Attributes to Adac Elements

Data to process

Model: design drainage data

Adac Attribute File: $LIB\adac_12d_[adac_version_by_ten].12duaf

Model for results: design 12d attributes data

Clean model first? ☑

Model <design drainage data> exists

model to process.

12duaf file to use - supplied by 12d Solutions

model for the updated ADAC Assets.

first clean the model

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 1.4.8.6 Apply User Attributes to ADAC Elements

Continue to the next section 1.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.6 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 updated 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

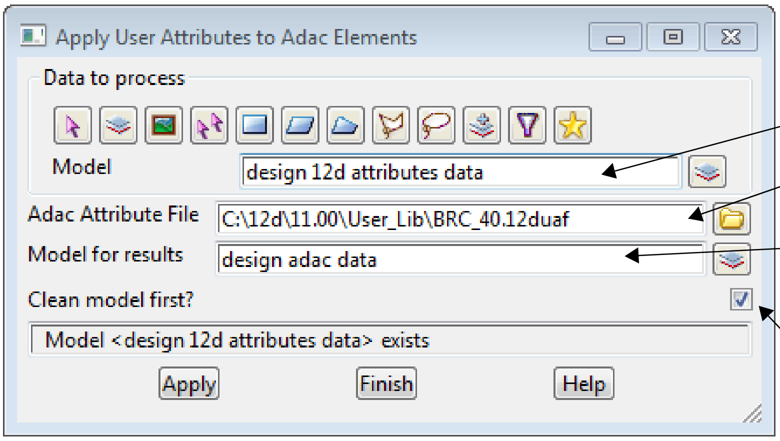
ADAC User Attributes to ADAC Element

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**.


Apply User Attributes to Adac Elements

Data to process

Model design 12d attributes data

Adac Attribute File C:\12d\11.00\User_Lib\BRC_40.12duaf

Model for results design adac data

Clean model first? ☑

Model <design 12d attributes data> exists

model to process.

company supplied 12duaf file to use

model for the updated ADAC Assets.

first clean the model

The model **design user attributes data** is on the view called **design string properties data.**

Note - this option is described in more detail in 1.4.8.6 Apply User Attributes to ADAC Elements

Continue to the next section 1.5.2.2.7 Update ADAC ObjectId - Design or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.7 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.

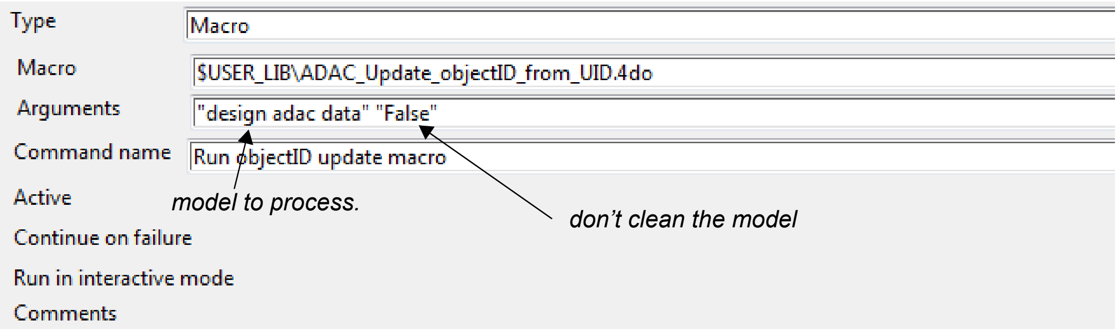
This is only relevant for designers and not surveyors.

(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	
Continue on failure	
Run in interactive mode	
Comments	

model to process.

don't clean the model

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 1.4.9.3.6.5 Set ADAC ObjectId from String UID .

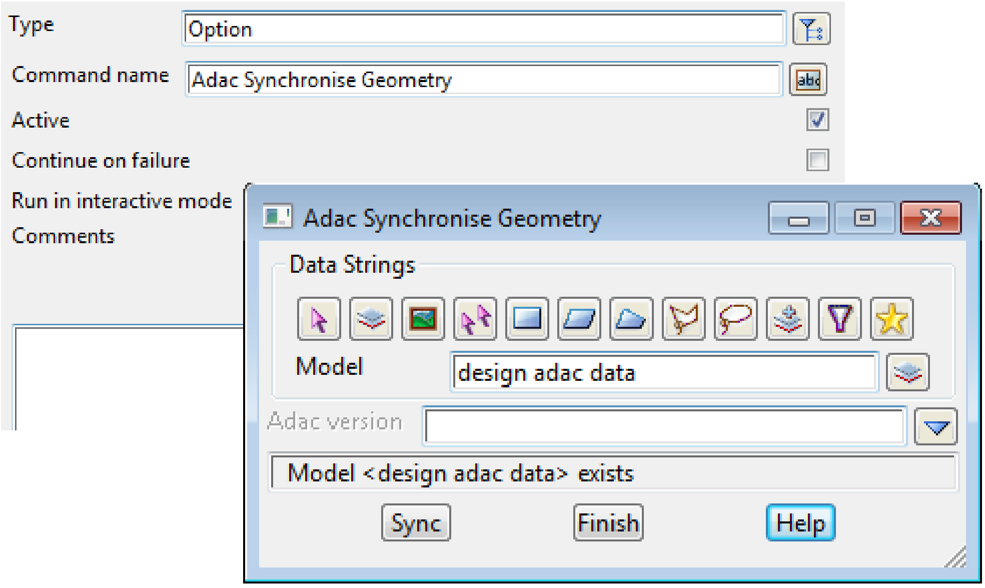
Continue to the next section 1.5.2.2.8 Update ADAC Geometry - Design or return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.5.2.2.8 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



Type: Option

Command name: Adac Synchronise Geometry

Active ☑

Continue on failure ☐

Run in interactive mode

Comments

Adac Synchronise Geometry

Data Strings

Model: design adac data

Adac version:

Model <design adac data> exists

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 1.4.8.4 Sync Geometry

Return to 1.5.2.2 ADAC Design Base Chain or 1.5.2 Design to ADAC Chains .

1.6 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

1.6.1 Setting Up Map Files for ADAC

1.6.2 What Data Prep is Needed for ADAC

1.6.3.2 Setting Up Your User ADAC Menu

1.6.4 Setting Up ADAC Templates

1.6.5 Setting Up Your User Keys

1.6.1 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

1.6.1.1 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**.

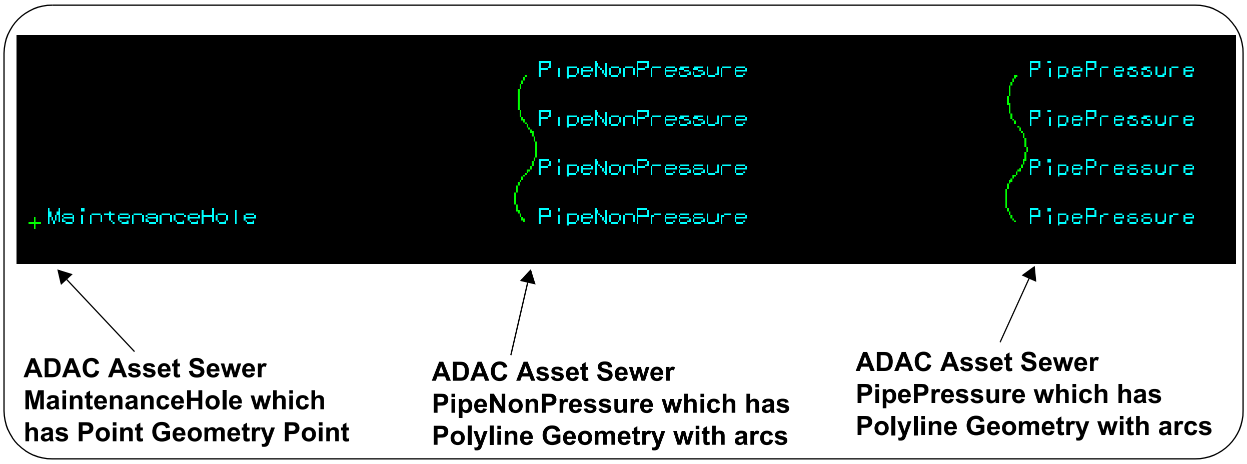
Or that a **Transport>RoadEdges>RoadEdge** must have a **Type** and it can only be one of B1, B2, B3, B4, B5, SM1, SM2, SM3,SM4,SM5, M1, M2, M3, M4, M5, M6, ER1, ER2, ER3, ER4, ER5, INV660, INV90o0, Bitumen and Concrete (as per DMR drawing 1033 - now TMR).

In all cases, any variations from the ADAC Schema will make it an invalid ADAC XML file.

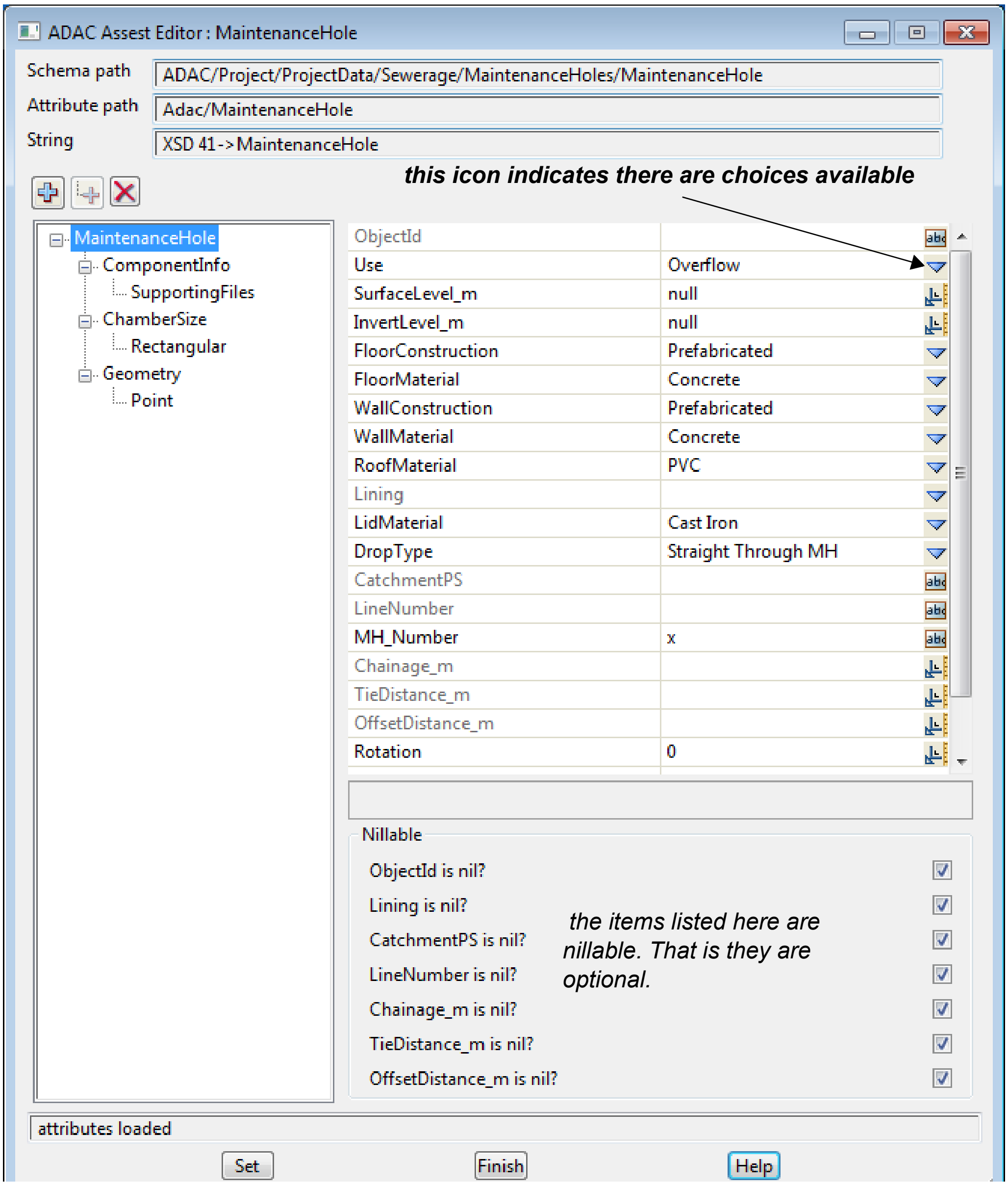
The authorised version of **ADAC Asset Design and As Constructed Schema Help File** (for Version 4.1.0 or Version 4.0.0) is controlled by the **IPWEA** and available on their web site www.engicom.com.au/products/adac2/. This is read to learn what the ADAC Assets all are.

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 it 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 1.4.8.3 XSD to Model .

For example, the first three strings in that model are:


PipeNonPressure
PipeNonPressure
PipeNonPressure
PipeNonPressure
PipePressure
PipePressure
PipePressure
PipePressure
MaintenanceHole
ADAC Asset Sewer MaintenanceHole which has Point Geometry Point
ADAC Asset Sewer PipeNonPressure which has Polyline Geometry with arcs
ADAC Asset Sewer PipePressure which has Polyline Geometry with arcs

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.*



ADAC Assest Editor : MaintenanceHole

Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole

Attribute path: Adac/MaintenanceHole

String: XSD 41->MaintenanceHole

this icon indicates there are choices available

ObjectId	
Use	Overflow
SurfaceLevel_m	null
InvertLevel_m	null
FloorConstruction	Prefabricated
FloorMaterial	Concrete
WallConstruction	Prefabricated
WallMaterial	Concrete
RoofMaterial	PVC
Lining	
LidMaterial	Cast Iron
DropType	Straight Through MH
CatchmentPS	
LineNumber	
MH_Number	x
Chainage_m	
TieDistance_m	
OffsetDistance_m	
Rotation	0

Nillable

ObjectId is nil? ☑

Lining is nil? ☑

CatchmentPS is nil? ☑

LineNumber is nil? ☑

Chainage_m is nil? ☑

TieDistance_m is nil? ☑

OffsetDistance_m is nil? ☑

the items listed here are nillable. That is they are optional.

attributes loaded

Continue to the next section 1.6.1.2 Know What Your Client Wants in the ADAC XML File or return to 1.6.1 Setting Up Map Files for ADAC .

1.6.1.2 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 1.6.1.3 How the ADAC Map File is Used or return to 1.6.1 Setting Up Map Files for ADAC .

1.6.1.3 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.

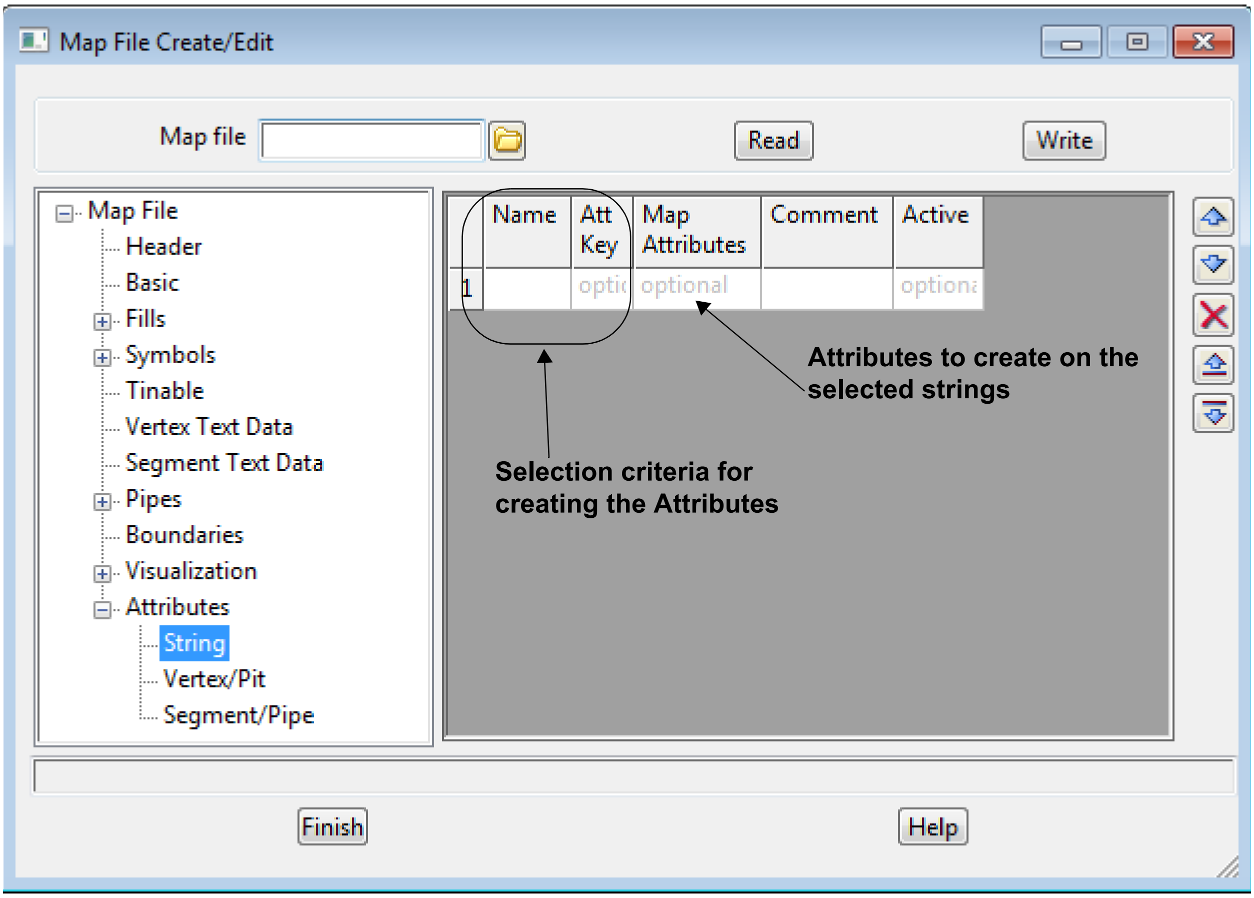
If the criteria are satisfied by a string, then the **Attributes** in the **Map Attributes** column are applied as string/vertex/segment attributes to that string. The next string is then processed.

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**.



	Name	Att Key	Map Attributes	Comment	Active
1		optic	optional		optiona

Attributes to create on the selected strings

Selection criteria for creating the Attributes

Apart from drainage and sewer strings, this is the only way strings are marked with and *ADAC Asset* by a ***Map File***.

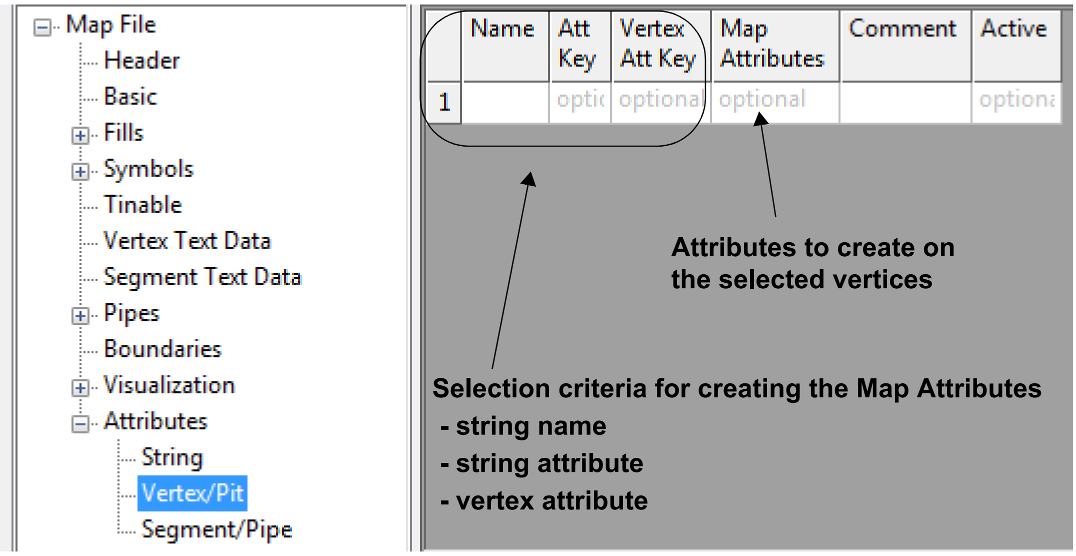
The string names are those created during the Design process, or for Surveyors from the field codes when a survey is reduced in ***12d Model***.

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		optic	optional	optional		optiona

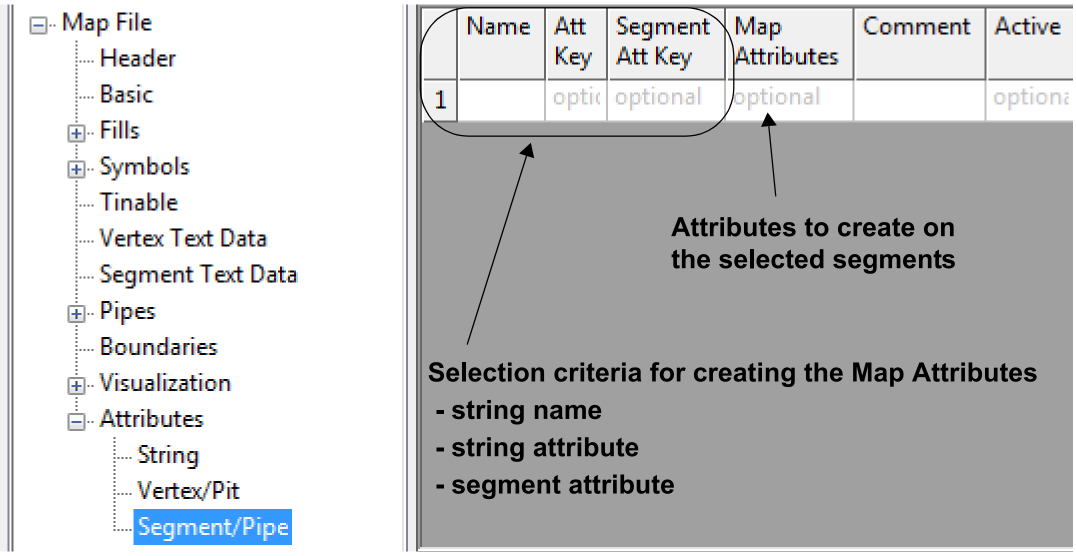
Attributes to create on the selected vertices

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		optic	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 1.6.1.4 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 1.6.1.6 Setting Up the Map File Selection Criteria

Continue to the next section 1.6.1.4 Creating the ADAC Attributes for Map Files or return to 1.6.1 Setting Up Map Files for ADAC .

1.6.1.4 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 nillability 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 1.4.8.2 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

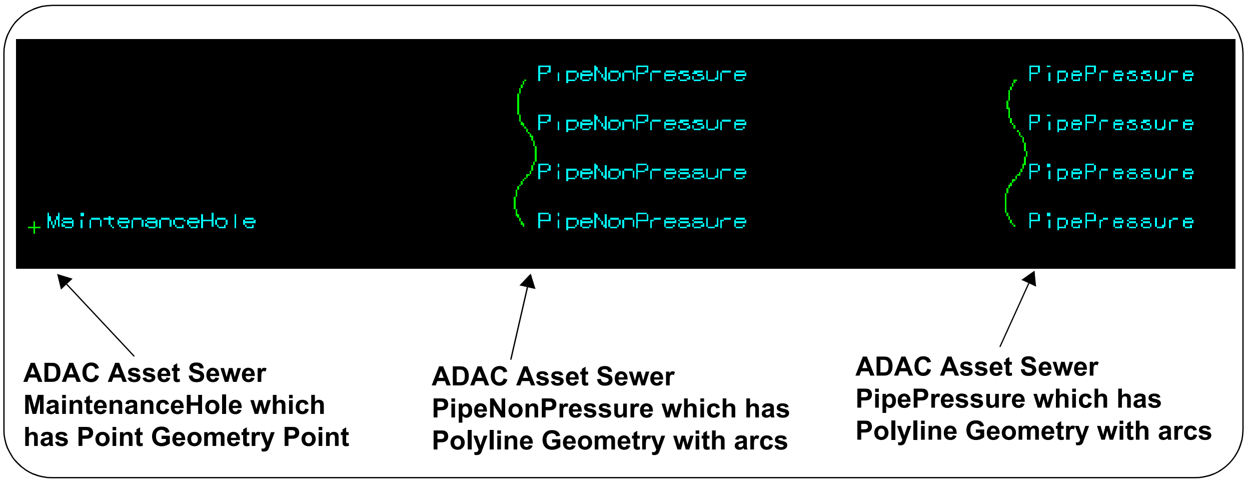
is used to take a user specified version of the *ADAC XML Schema* XSD and produces a model with a 12d string for each of the ADAC Assets (so sixty eight strings) and each string has

(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 1.4.8.3 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:


PipeNonPressure
PipeNonPressure
PipeNonPressure
PipeNonPressure
PipePressure
PipePressure
PipePressure
PipePressure
MaintenanceHole
ADAC Asset Sewer MaintenanceHole which has Point Geometry Point
ADAC Asset Sewer PipeNonPressure which has Polyline Geometry with arcs
ADAC Asset Sewer PipePressure which has Polyline Geometry with arcs

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 (1.4.3 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.

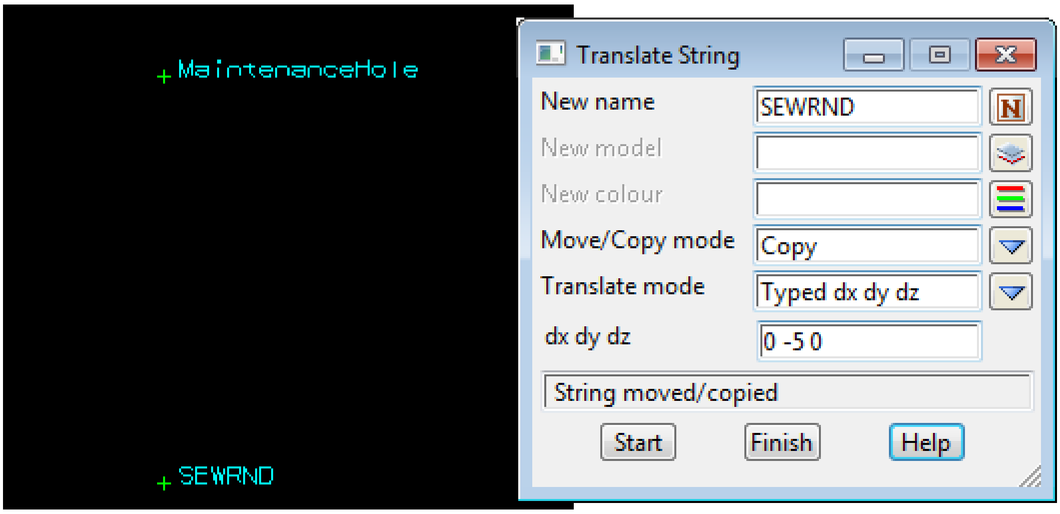
When you run **ADAC Strings to Map File**, each ADAC string creates an entry in either the *Attributes>String*, *Attributes>Vertex/Pit* or *Attributes>Segment/Pipe* section of the ***Map File*** (see 1.4.8.1 ADAC Strings to Map File).

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.



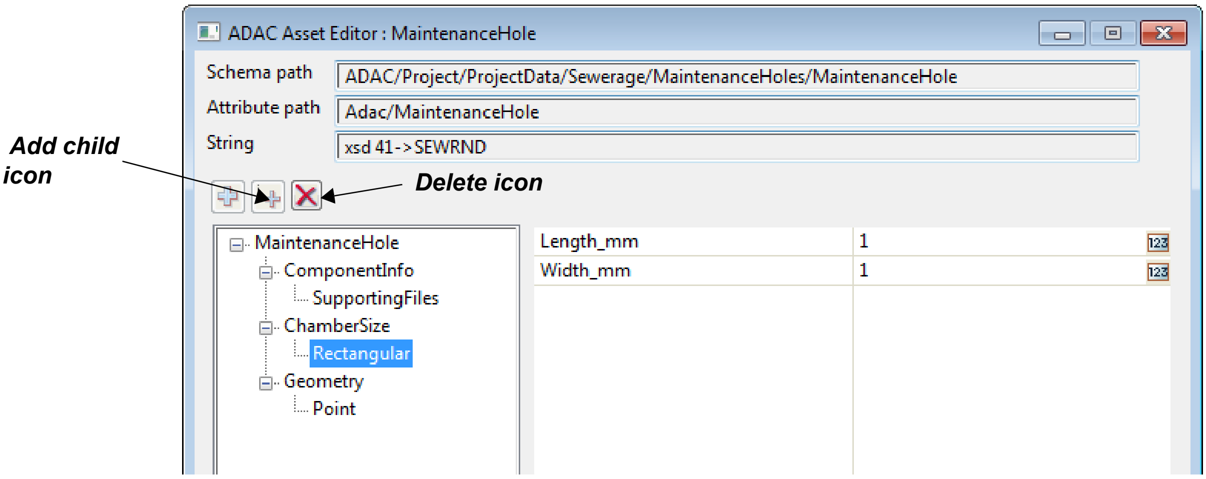
New name: SEWRND
New model:
New colour:
Move/Copy mode: Copy
Translate mode: Typed dx dy dz
dx dy dz: 0 -5 0
String moved/copied

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.

-

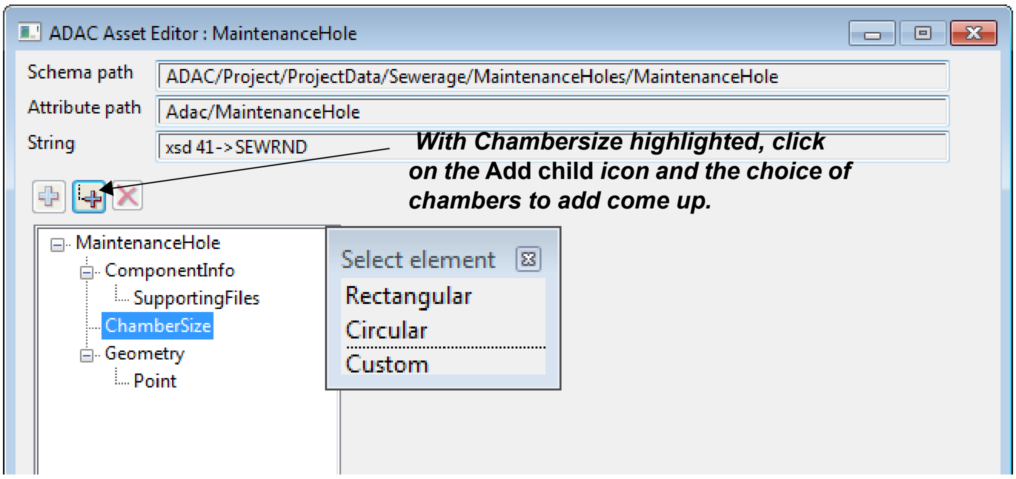


Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole
Attribute path: Adac/MaintenanceHole
String: xsd 41->SEWRND

Length_mm	1
Width_mm	1

Then click on the **Add Child** icon and select *Circular* from the **Select Element** panel that pops up.

-



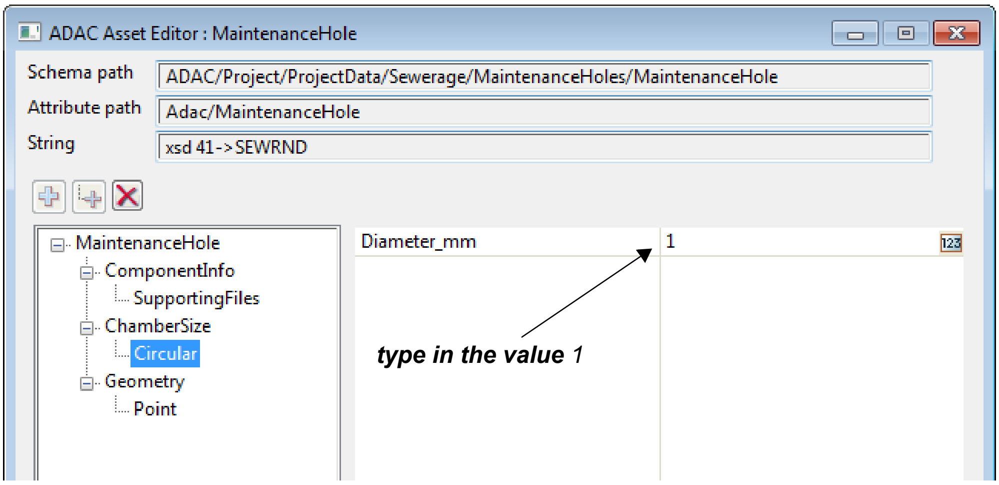
Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole
Attribute path: Adac/MaintenanceHole
String: xsd 41->SEWRND

With Chambersize highlighted, click on the Add child icon and the choice of chambers to add come 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.



ADAC Asset Editor : MaintenanceHole

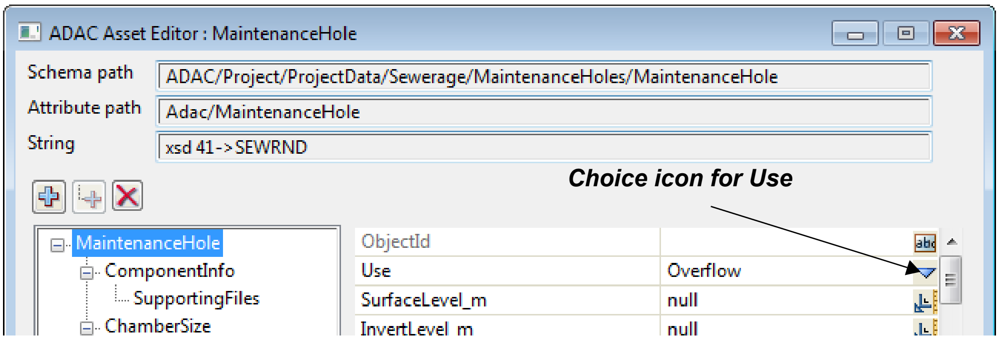
Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole
Attribute path: Adac/MaintenanceHole
String: xsd 41->SEWRND

- MaintenanceHole
  - ComponentInfo
    - SupportingFiles
  - ChamberSize
    - Circular
  - Geometry
    - Point

Diameter_mm	1

type in the value 1

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*.



ADAC Asset Editor : MaintenanceHole

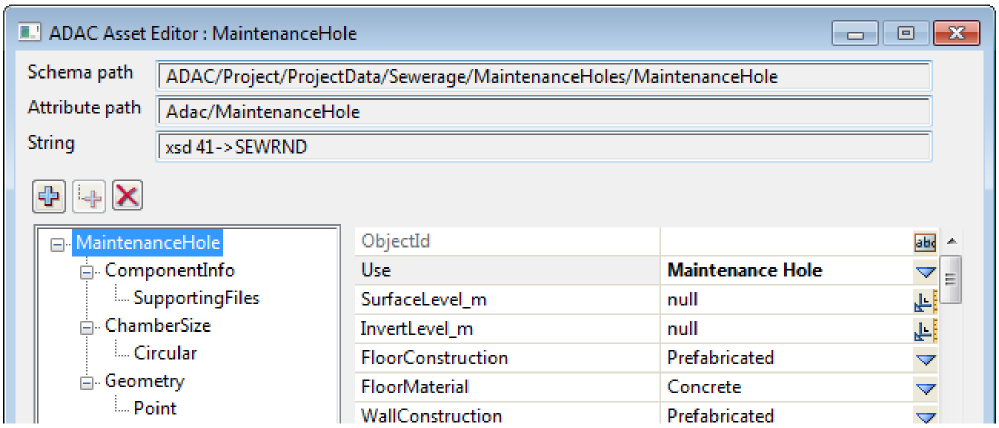
Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole
Attribute path: Adac/MaintenanceHole
String: xsd 41->SEWRND

Choice icon for Use

- MaintenanceHole
  - ComponentInfo
    - SupportingFiles
  - ChamberSize

ObjectId	
Use	Overflow
SurfaceLevel_m	null
InvertLevel_m	null

Click on the **Choice** icon for **Use** and choose *MaintenanceHole* from the list.



ADAC Asset Editor : MaintenanceHole

Schema path: ADAC/Project/ProjectData/Sewerage/MaintenanceHoles/MaintenanceHole
Attribute path: Adac/MaintenanceHole
String: xsd 41->SEWRND

- MaintenanceHole
  - ComponentInfo
    - SupportingFiles
  - ChamberSize
    - Circular
  - Geometry
    - Point

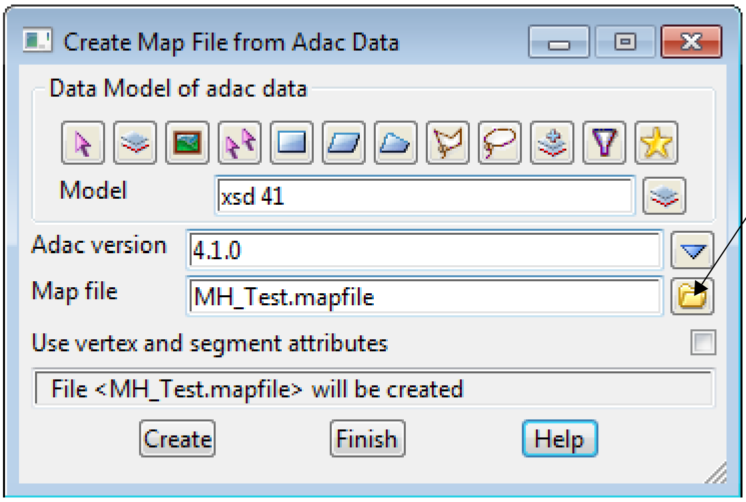
ObjectId	
Use	**Maintenance Hole**
SurfaceLevel_m	null
InvertLevel_m	null
FloorConstruction	Prefabricated
FloorMaterial	Concrete
WallConstruction	Prefabricated

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:



Model: xsd 41

Adac version: 4.1.0

Map file: MH_Test.mapfile

Use vertex and segment attributes □

File <MH_Test.mapfile> will be created

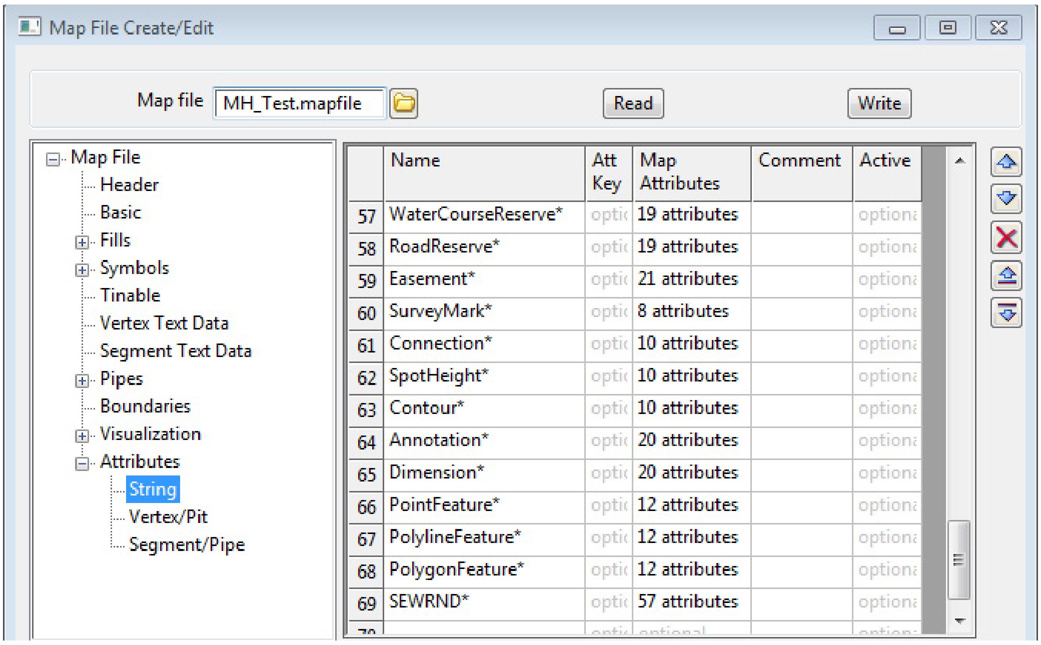
click on the folder icon once the mapfile is created and select Open to edit the Map File

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.



Map file: MH_Test.mapfile

	Name	Att Key	Map Attributes	Comment	Active
57	WaterCourseReserve*	optic	19 attributes		optiona
58	RoadReserve*	optic	19 attributes		optiona
59	Easement*	optic	21 attributes		optiona
60	SurveyMark*	optic	8 attributes		optiona
61	Connection*	optic	10 attributes		optiona
62	SpotHeight*	optic	10 attributes		optiona
63	Contour*	optic	10 attributes		optiona
64	Annotation*	optic	20 attributes		optiona
65	Dimension*	optic	20 attributes		optiona
66	PointFeature*	optic	12 attributes		optiona
67	PolylineFeature*	optic	12 attributes		optiona
68	PolygonFeature*	optic	12 attributes		optiona
69	SEWRND*	optic	57 attributes		optiona

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 1.6.1.5 Notes On Creating the ADAC Attribute Structure or return to 1.6.1 Setting Up Map Files for ADAC .

1.6.1.5 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 1.4.8.1 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 1.6.1.6 Setting Up the Map File Selection Criteria or return to 1.6.1 Setting Up Map Files for ADAC .

1.6.1.6 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 1.6.1.6.1 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 1.6.1.6.2 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 1.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings

1.6.1.6.1 Using String Names to Select Strings

This is only for strings that are not Drainage or Sewer strings

See 1.6.1.6.1.1 Designers' Approach

See 1.6.1.6.1.2 Surveyor Approach - Using Survey Codes in the Field

1.6.1.6.1.1 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>RoadEges>RoadEdge** but it would not be enough to identify it as a **RoadEge** 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 1.6.1.6.2 Using String Name and/or String Attributes to Select Strings

1.6.1.6.1.2 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 1.6.1.6.2 Using String Name and/or String Attributes to Select Strings

1.6.1.6.2 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

1.6.1.6.2.1 Entering User Attributes in the Field

1.6.1.6.2.2 Running a 12d Option or 12dPL that Produce User Attributes

1.6.1.6.2.3 Manually Entering User Attributes

1.6.1.6.2.4 Using the Name and Att Key in the ADAC Map File

1.6.1.6.2.1 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 1.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey .

Continue to the next section 1.6.1.6.2.2 Running a 12d Option or 12dPL that Produce User Attributes or return to 1.6.1.6.2 Using String Name and/or String Attributes to Select Strings .

1.6.1.6.2.2 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, 1.4.9.2.3 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.

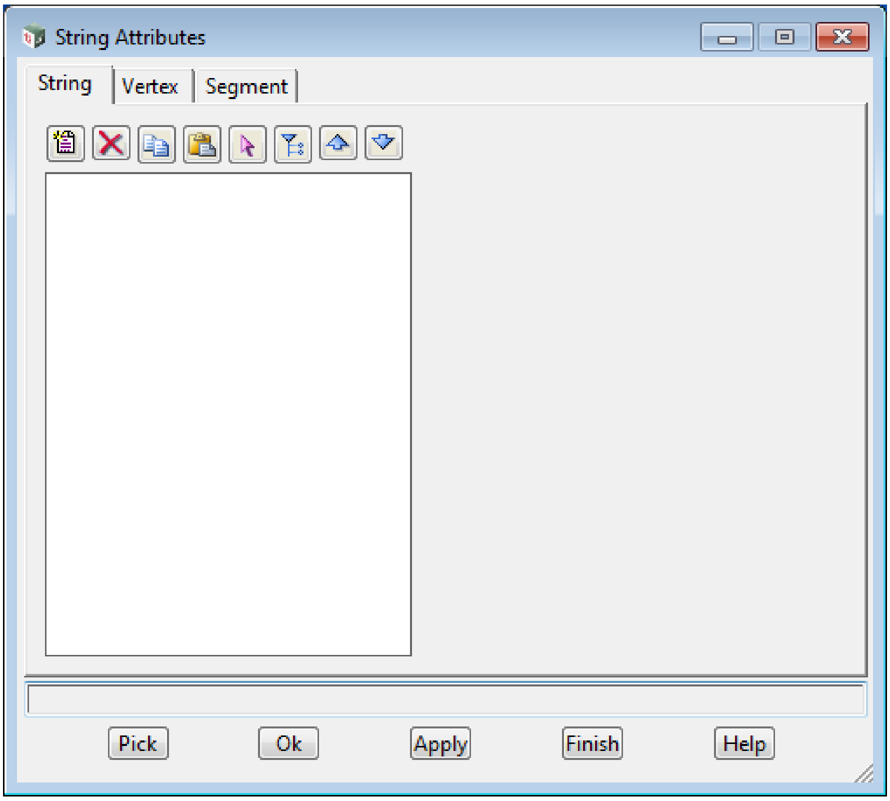
Continue to the next section 1.6.1.6.2.3 Manually Entering User Attributes or return to 1.6.1.6.2 Using String Name and/or String Attributes to Select Strings .

1.6.1.6.2.3 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 1.5.1.2.5 Update ADAC Elements from User Created Attributes on the String - Survey and 1.5.2.2.6 Update ADAC Elements from User Created Attributes - Design .

Continue to the next section 1.6.1.6.2.4 Using the Name and Att Key in the ADAC Map File or return to 1.6.1.6.2 Using String Name and/or String Attributes to Select Strings .

1.6.1.6.2.4 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 1.6.1.4 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.

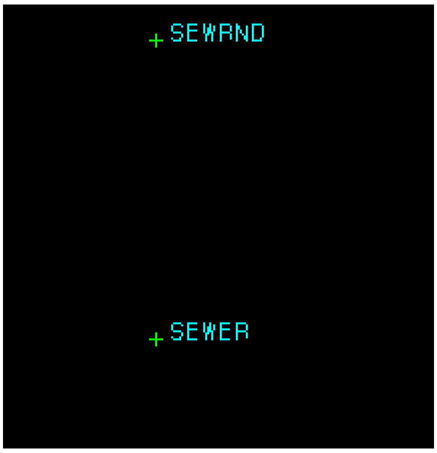
To explain the process of adding a string attribute that becomes an **Att Key**, we will build on the previous example and

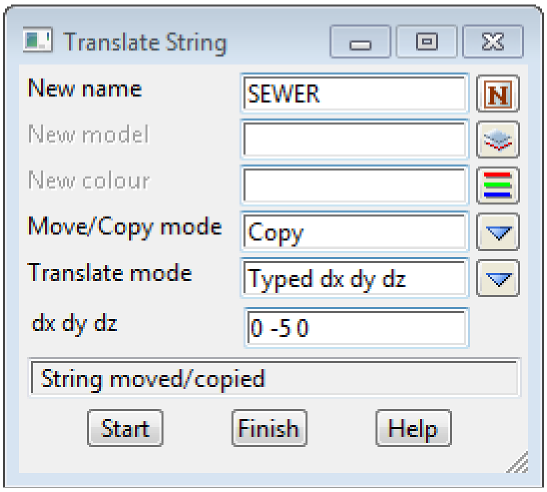
> Instead of having a **string name** SEWRND to indicate the *Sewerage>MaintenanceHoles>MaintenanceHole* has a circular chamber, we only have the **nondescript name** SEWER and a **string attribute** called **Chamber** which has the text value of **Circular** or **Rectangular**.

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**.





New name SEWER
New model
New colour
Move/Copy mode Copy
Translate mode Typed dx dy dz
dx dy dz 0 -5 0

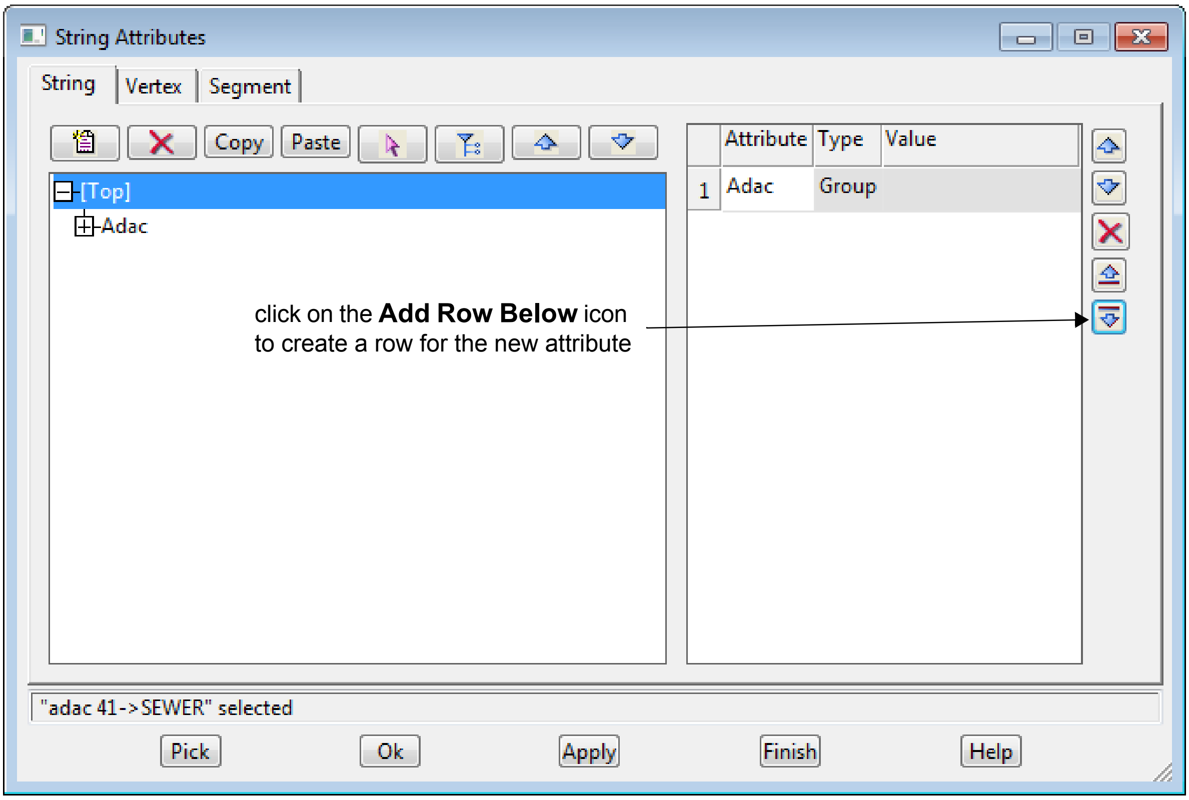
String moved/copied

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.

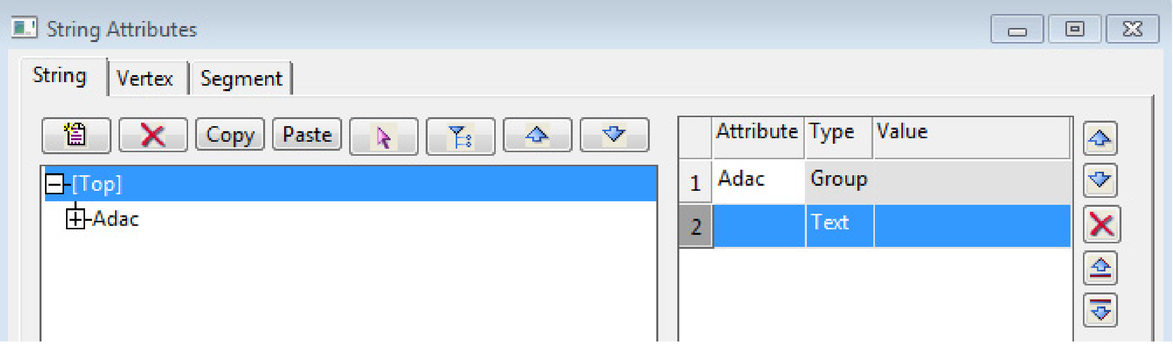


	Attribute	Type	Value
1	Adac	Group	

click on the **Add Row Below** icon to create a row for the new attribute

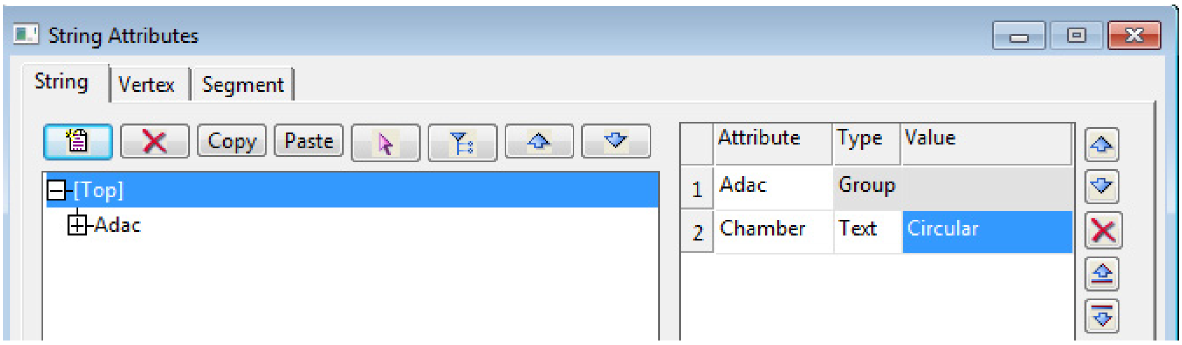
"adac 41->SEWER" selected

Highlight the node **[Top]** and then click on the **Add Row Below** icon to create a new row to use for the attribute **Circular**.



	Attribute	Type	Value
1	Adac	Group	
2		Text	

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.

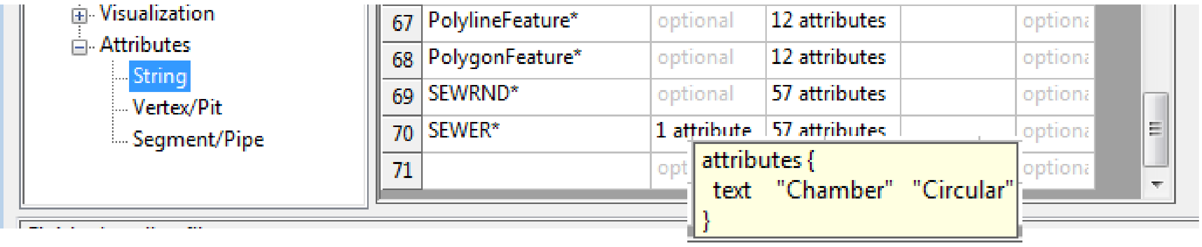


	Attribute	Type	Value
1	Adac	Group	
2	Chamber	Text	Circular

To create the ***Map File,*** we repeat the process used in 1.6.1.4 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.



67	PolylineFeature*	optional	12 attributes		optiona
68	PolygonFeature*	optional	12 attributes		optiona
69	SEWRND*	optional	57 attributes		optiona
70	SEWER*	1 attribute	57 attributes		optiona
71		opt			optiona

```
attributes {
 text "Chamber" "Circular"
}
```

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 1.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings or return to 1.6.1.6 Setting Up the Map File Selection Criteria .

1.6.1.6.3 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 **sewertype** of type Integer with the value **0** if it is a **drainage string**, or **1** if it a **sewer** string.

Being a **string attribute**, the attribute **sewertype** 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 **sewertype** of value **1** then it is a *Sewerage>MaintenanceHoles>MaintenanceHole*

(ii) vertex with an integer string attribute **sewertype** of value **0** then it is a *StormWater>Pits>Pit*

(iii) segment with an integer string attribute **sewertype** of value **1** then it is a *Sewerage>PipesNonPressure>PipeNonPresssure*.

(iv) segment with an integer string attribute **sewertype** of value **0** then it is a *StormWater>Pipes>Pipe*.

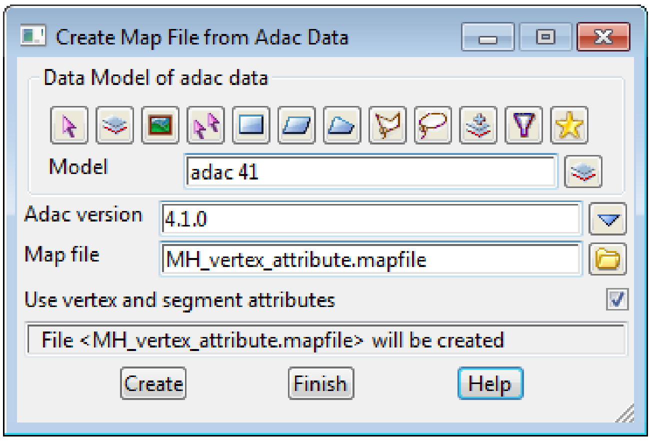
Important Note - Actually a ***Map File*** can't tell the difference between ANY string types, so the presence of a **string attribute** called **sewertype** with value **0** or **1** can be used in a ***Map File*** to select only a sewer string or a drainage string.

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,



Create Map File from Adac Data

Data Model of adac data

Model: adac 41

Adac version: 4.1.0

Map file: MH_vertex_attribute.mapfile

Use vertex and segment attributes ☑

File <MH_vertex_attribute.mapfile> will be created

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 1.6.1.4 Creating the ADAC Attributes for Map Files)

A string attribute on the ADAC Asset with 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

1.6.1.6.3.1 Setting Up the Attributes>Vertex Section of the Map File
1.6.1.6.3.2 Setting Up the Attributes>Segment Section of the Map File
1.6.1.6.3.3 Bonuses for Drainage and Sewer Strings

1.6.1.6.3.1 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**.

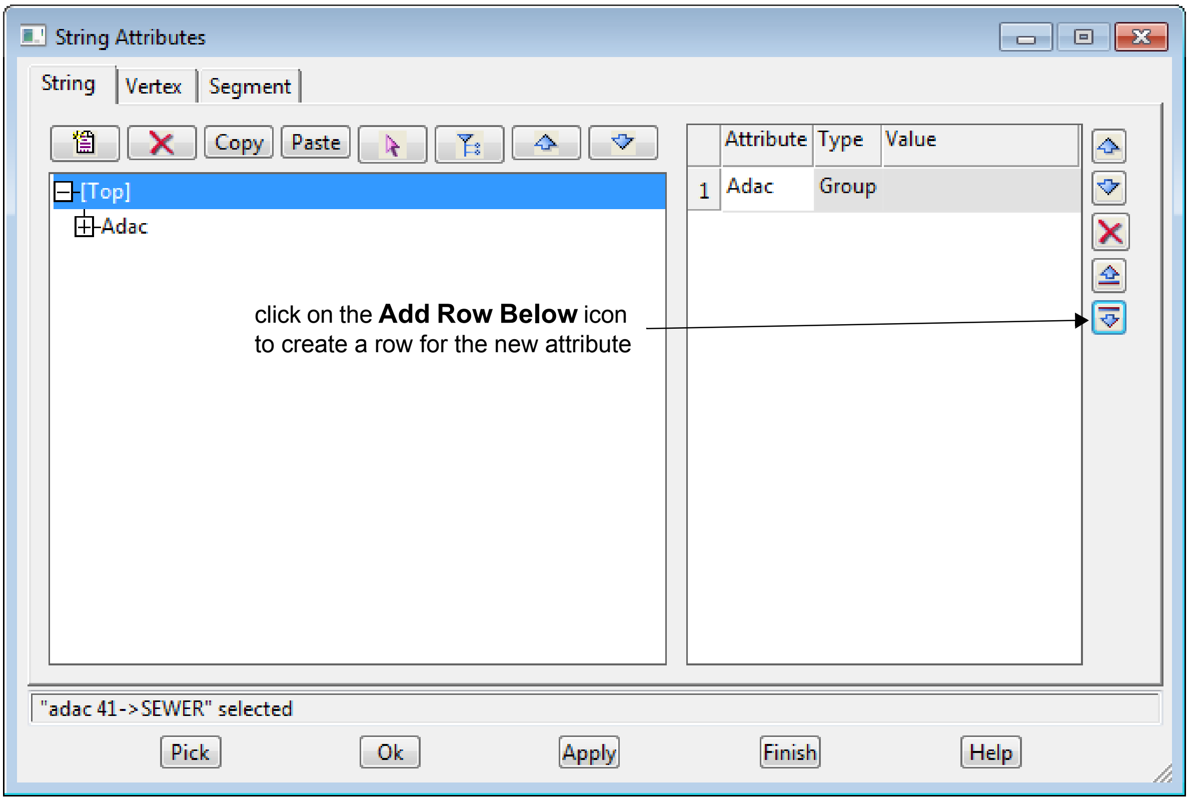
Note: To allow the user to identify which *ADAC Asset* string creates a particular row in the *Attributes>Vertex/Pit* grid, the string **name** is copied into the **Comment** column.

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.



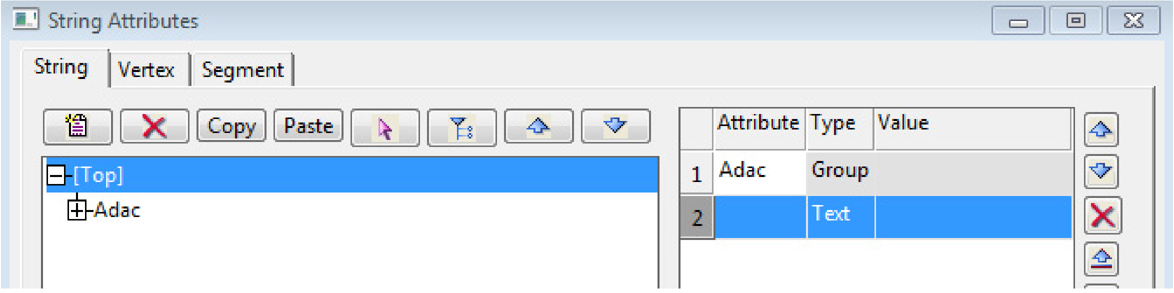
	Attribute	Type	Value
1	Adac	Group	

click on the **Add Row Below** icon to create a row for the new attribute

"adac 41->SEWER" selected

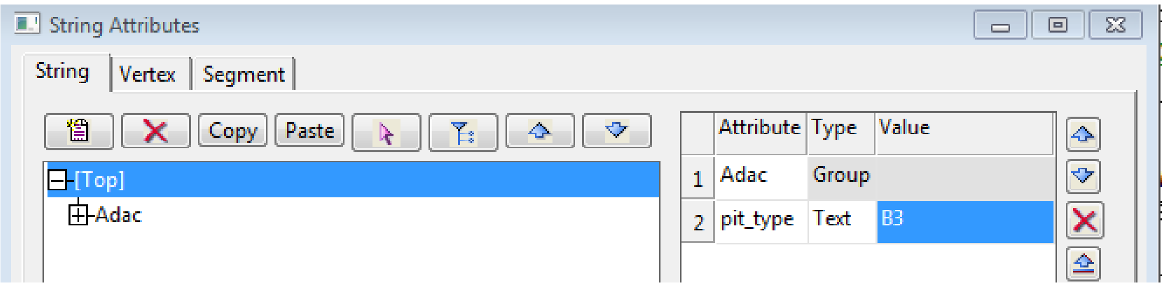
For the **String** tab, highlight the node **[Top]** and then click on the **Add Row Below** icon to create a new row to use for the attribute **pit_type**.

Note: if there are other first level attributes other than ADAC, highlight delete them.



	Attribute	Type	Value
1	Adac	Group	
2		Text	

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.

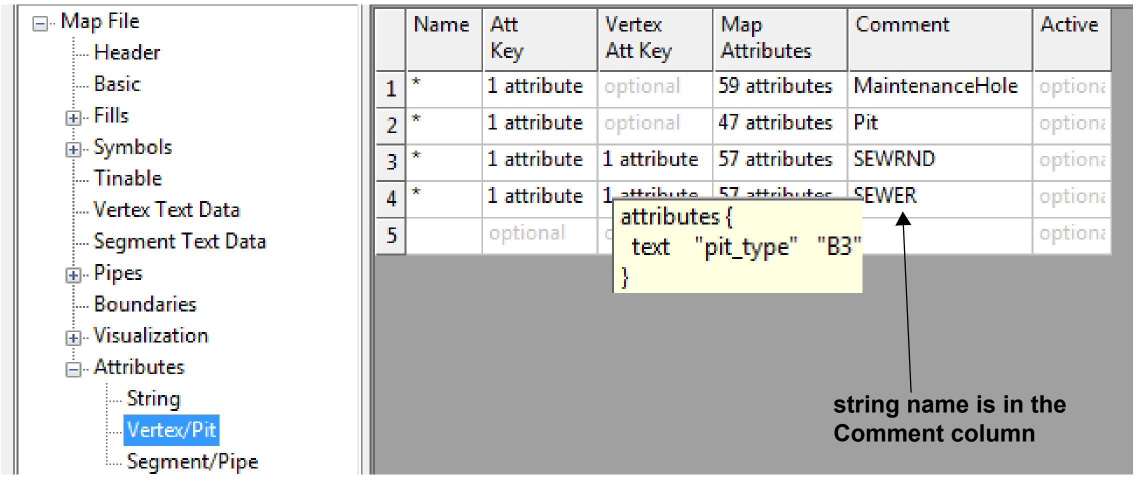


	Attribute	Type	Value
1	Adac	Group	
2	pit_type	Text	B3

To create the ***Map File,*** we repeat the process used in 1.6.1.4 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.



	Name	Att Key	Vertex Att Key	Map Attributes	Comment	Active
1	*	1 attribute	optional	59 attributes	MaintenanceHole	optiona
2	*	1 attribute	optional	47 attributes	Pit	optiona
3	*	1 attribute	1 attribute	57 attributes	SEWRND	optiona
4	*	1 attribute	1 attribute	57 attributes	SEWER	optiona
5		optional				optiona

```
attributes {
  text   "pit_type"   "B3"
}
```

string name is in the Comment column

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 **sewertype** is **0** and the ADAC Asset attributes to use are those of a *StormWater>Pits>Pit*.

Continue to the next section 1.6.1.6.3.2 Setting Up the Attributes>Segment Section of the Map File or return to 1.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings .

1.6.1.6.3.2 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**.

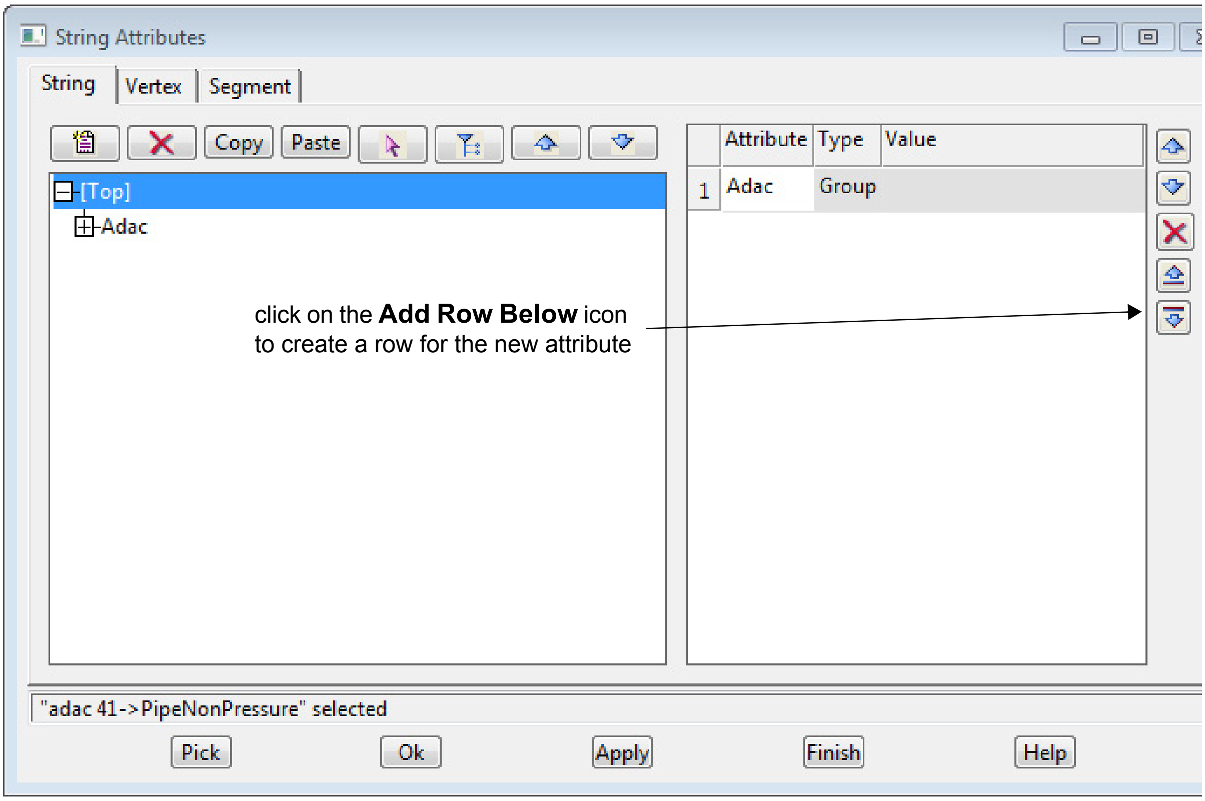
Note: To allow the user to identify which *ADAC Asset* string creates a particular row in the *Attributes>Segment/Pipe* grid, the string **name** is copied into the **Comment** column.

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**.



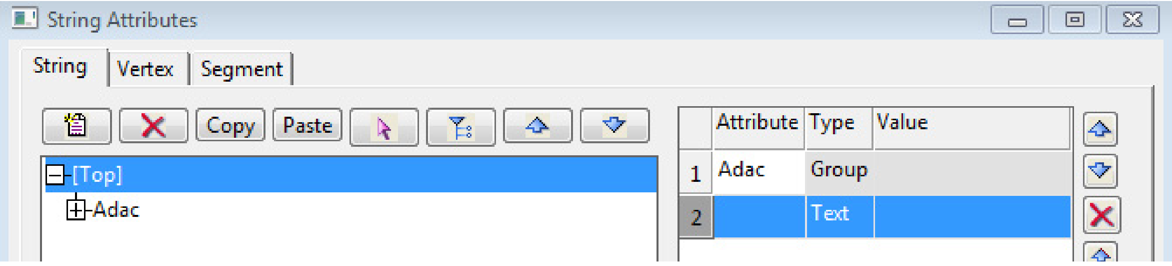
	Attribute	Type	Value
1	Adac	Group	

click on the **Add Row Below** icon to create a row for the new attribute

"adac 41->PipeNonPressure" selected

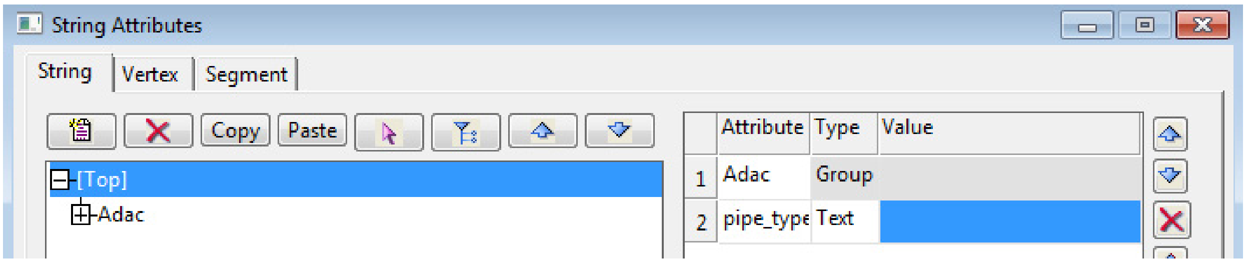
For the **String** tab, highlight the node **[Top]** and then click on the **Add Row Below** icon to create a new row to use for the attribute **pipe_type**.

Note: if there are other first level attributes other than ADAC, highlight delete them.



	Attribute	Type	Value
1	Adac	Group	
2		Text	

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.

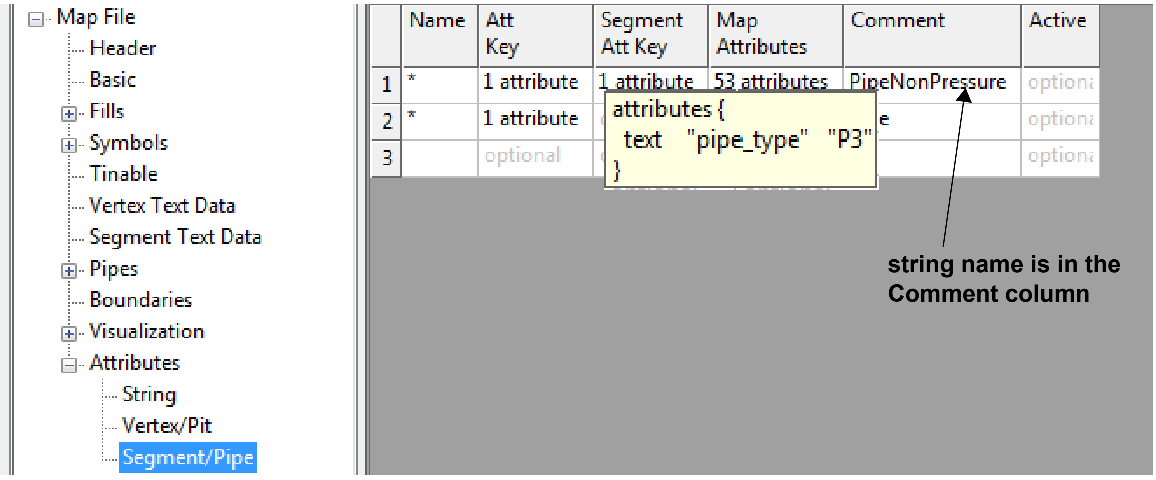


	Attribute	Type	Value
1	Adac	Group	
2	pipe_type	Text	

To create the ***Map File,*** we repeat the process used in 1.6.1.4 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.



	Name	Att Key	Segment Att Key	Map Attributes	Comment	Active
1	*	1 attribute	1 attribute	53 attributes	PipeNonPressure	optiona
2	*	1 attribute			e	optiona
3		optional				optiona

attributes {
 text "pipe_type" "P3"
}

string name is in the Comment column

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 **sewertype** is **0** and the ADAC Asset attributes to use are those of a *StormWater>Pipes>Pipe*.

Continue to the next section 1.6.1.6.3.3 Bonuses for Drainage and Sewer Strings or return to 1.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings .

1.6.1.6.3.3 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 1.5.2.2.4 Update ADAC Elements from Drainage/Sewer String Properties .

Continue to the next section 1.6.1.7 More on Creating the ADAC Map Files or return to 1.6.1.6.3 Using Vertices and Segments - Drainage/Sewer Strings or 1.6.1.6 Setting Up the Map File Selection Criteria .

1.6.1.7 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 1.6.2 What Data Prep is Needed for ADAC or return to 1.6.1 Setting Up Map Files for ADAC .

1.6.2 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 1.4.9.2 Data Prep .

Continue to the next section 1.6.3 Setting up and Running ADAC Chains from the Menus or return to 1.6 Setting Up for ADAC .

1.6.3 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 1.6.3.1 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.

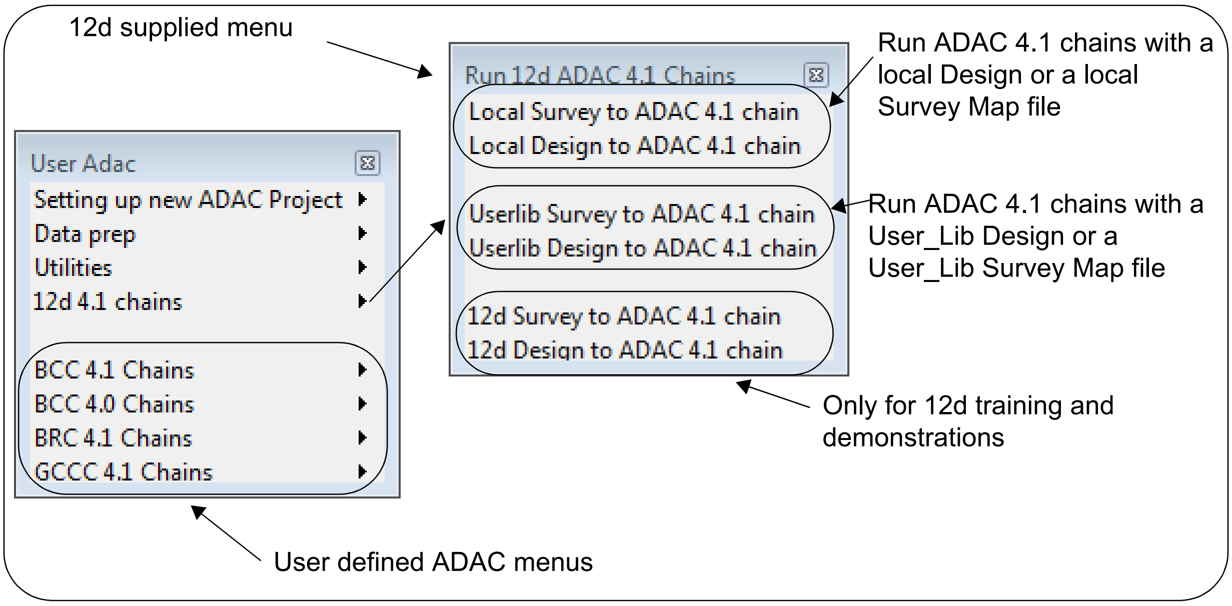
For information on setting up your own options on the **User ADAC** menu, see 1.6.3.2 Setting Up Your User ADAC Menu .

Or return to 1.6 Setting Up for ADAC .

1.6.3.1 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.


12d supplied menu
Run 12d ADAC 4.1 Chains
Local Survey to ADAC 4.1 chain
Local Design to ADAC 4.1 chain
Userlib Survey to ADAC 4.1 chain
Userlib Design to ADAC 4.1 chain
12d Survey to ADAC 4.1 chain
12d Design to ADAC 4.1 chain
Run ADAC 4.1 chains with a local Design or a local Survey Map file
Run ADAC 4.1 chains with a User_Lib Design or a User_Lib Survey Map file
Only for 12d training and demonstrations
User Adac
Setting up new ADAC Project
Data prep
Utilities
12d 4.1 chains
BCC 4.1 Chains
BCC 4.0 Chains
BRC 4.1 Chains
GCCC 4.1 Chains
User defined ADAC menus

(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 1.6.3.2 Setting Up Your User ADAC Menu or return to 1.6 Setting Up for ADAC .

1.6.3.2 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 there 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.

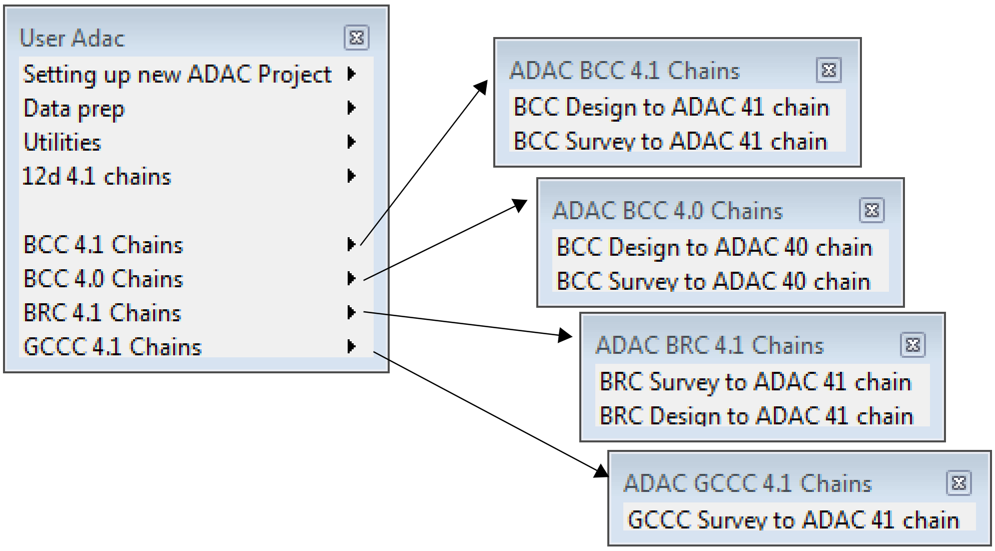
GCCC are using ADAC 4.1 only and at this stage and you are only providing them with Survey ADAC XML files.

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:

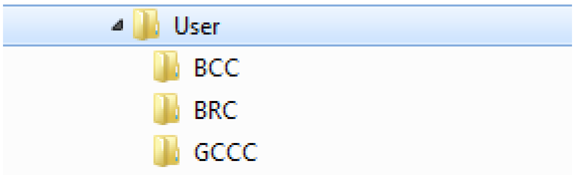

User Adac
Setting up new ADAC Project
Data prep
Utilities
12d 4.1 chains
BCC 4.1 Chains
BCC 4.0 Chains
BRC 4.1 Chains
GCCC 4.1 Chains
ADAC BCC 4.1 Chains
BCC Design to ADAC 41 chain
BCC Survey to ADAC 41 chain
ADAC BCC 4.0 Chains
BCC Design to ADAC 40 chain
BCC Survey to ADAC 40 chain
ADAC BRC 4.1 Chains
BRC Survey to ADAC 41 chain
BRC Design to ADAC 41 chain
ADAC GCCC 4.1 Chains
GCCC Survey to ADAC 41 chain

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 **User\BCC**) 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*, *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) adac_attribute_file is
$USER_LIB\[company]_[adac_version_by_ten].12duaf

(ix) adac_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>"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***.

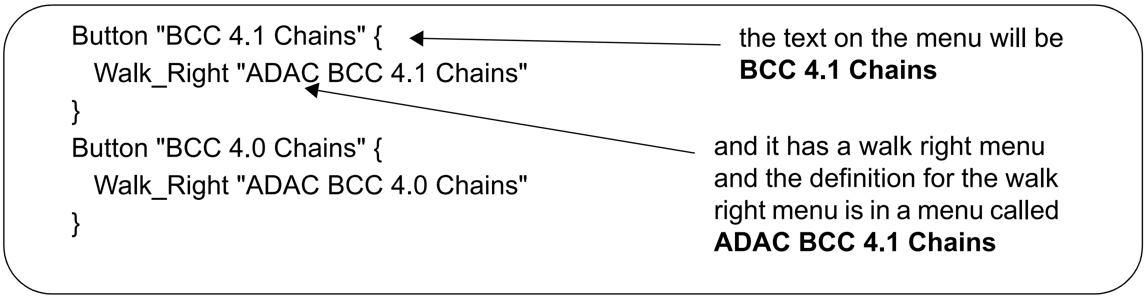
(d) Inside each customer subfolder (for example, inside **User\BCC**) there must be a file called ***ADAC_customer_user_menu.4d*** which creates the walk right menus for that customer that go on your **User** menu.

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:

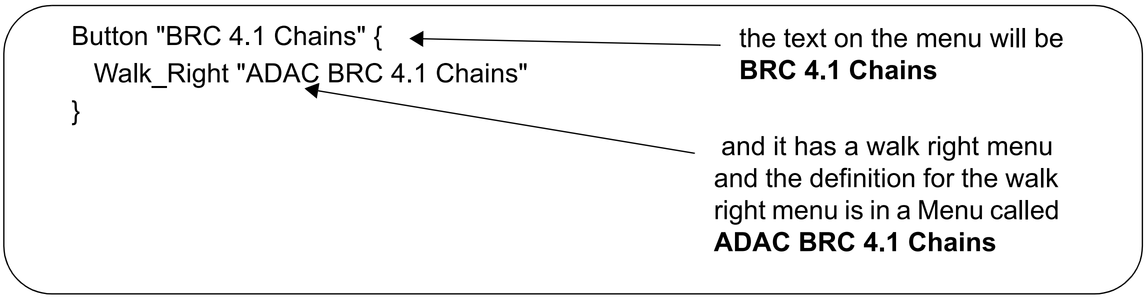


```
Button "BCC 4.1 Chains" {
  Walk_Right "ADAC BCC 4.1 Chains"
}
Button "BCC 4.0 Chains" {
  Walk_Right "ADAC BCC 4.0 Chains"
}
```

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.

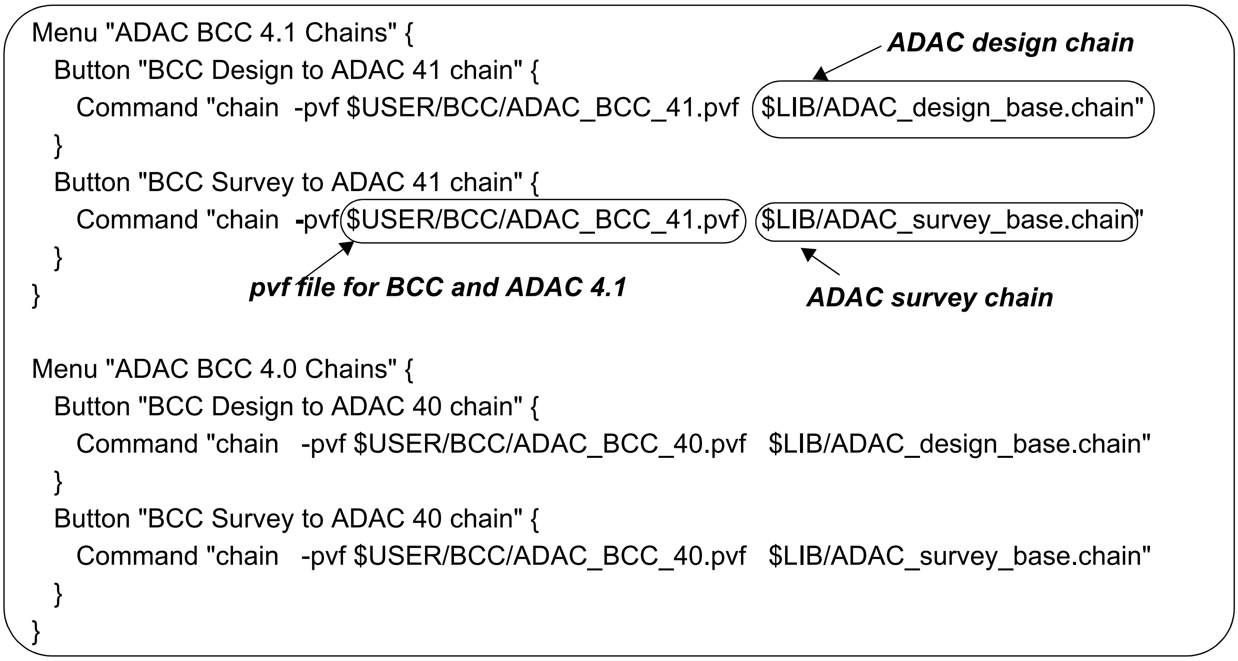
So now we need to define walk-right menus mentioned in the ***ADAC_customer_user_menu.4d*** files.

(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.

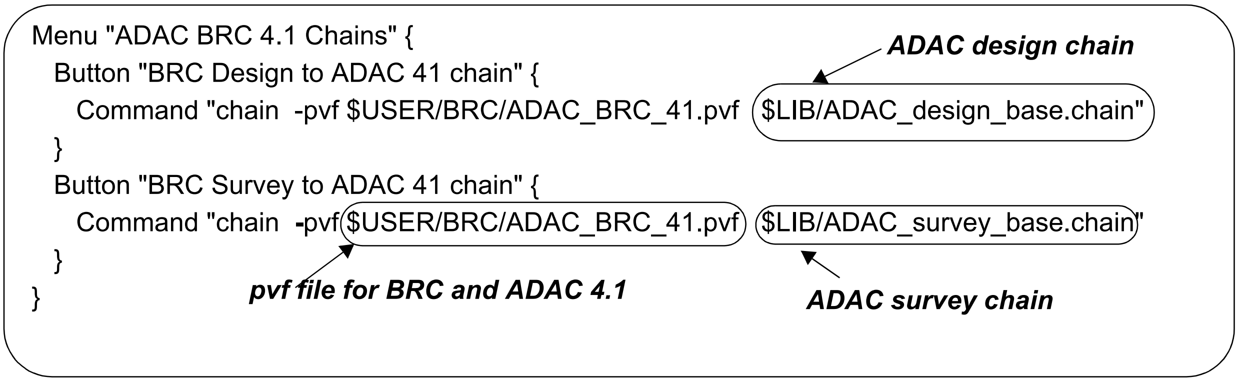
In our example, **BCC** has tow walk right menus and they both need options to run Design and Survey chains.

So in the folder **User\BCC**, the contents of ***ADAC_customer_walk_right_menu.4d*** are



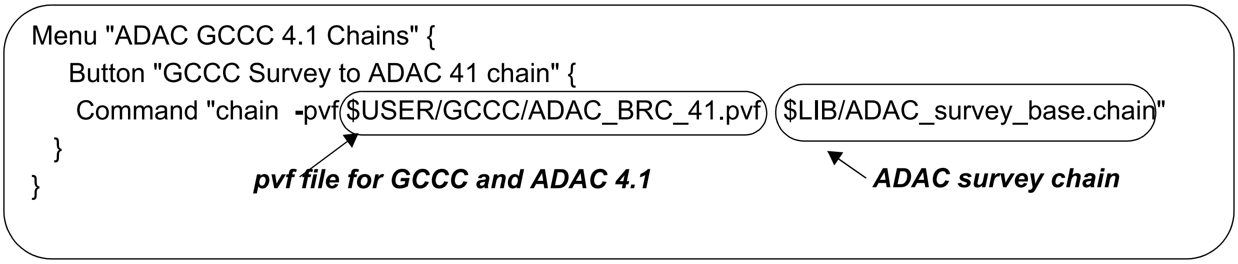
```
Menu "ADAC BCC 4.1 Chains" {
  Button "BCC Design to ADAC 41 chain" {
    Command "chain  -pvf $USER/BCC/ADAC_BCC_41.pvf  $LIB/ADAC_design_base.chain"
  }
  Button "BCC Survey to ADAC 41 chain" {
    Command "chain  -pvf $USER/BCC/ADAC_BCC_41.pvf  $LIB/ADAC_survey_base.chain"
  }
}

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"
  }
}
```

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"
  }
}
```

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"
  }
}
```

(f) Finally the ***ADAC_customer_user_menu.4d***'s for each of the customers needs to be included in the **ADAC User** menu

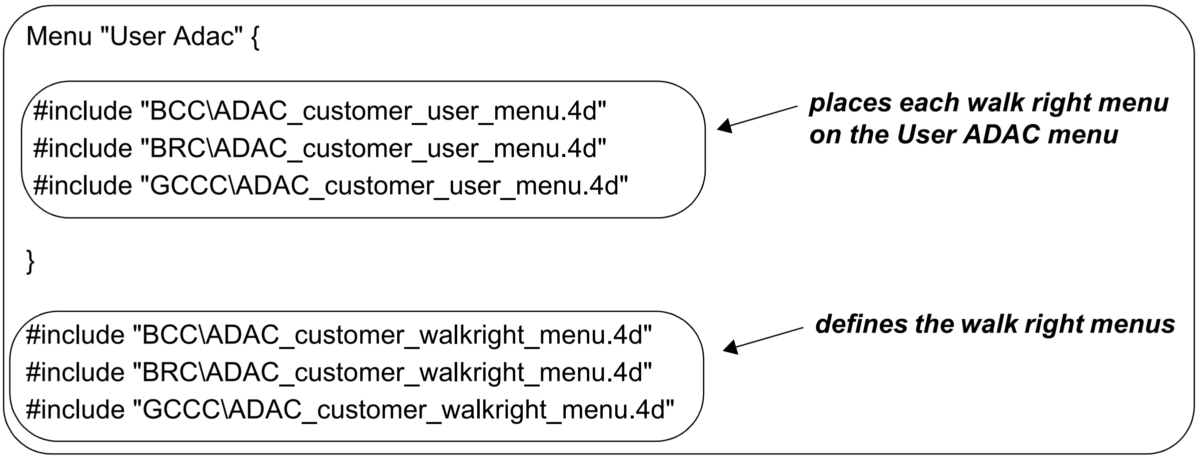
This is done by having a file in the **User** folder called ***ADAC_customer_user_menu.4d***

(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 1.6.4 Setting Up ADAC Templates or return to 1.6.3 Setting up and Running ADAC Chains from the Menus or 1.6 Setting Up for ADAC .

1.6.4 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

1.6.4.1 ADAC Header Templates

1.6.4.2 ADAC Asset Templates

1.6.4.1 ADAC Header Templates

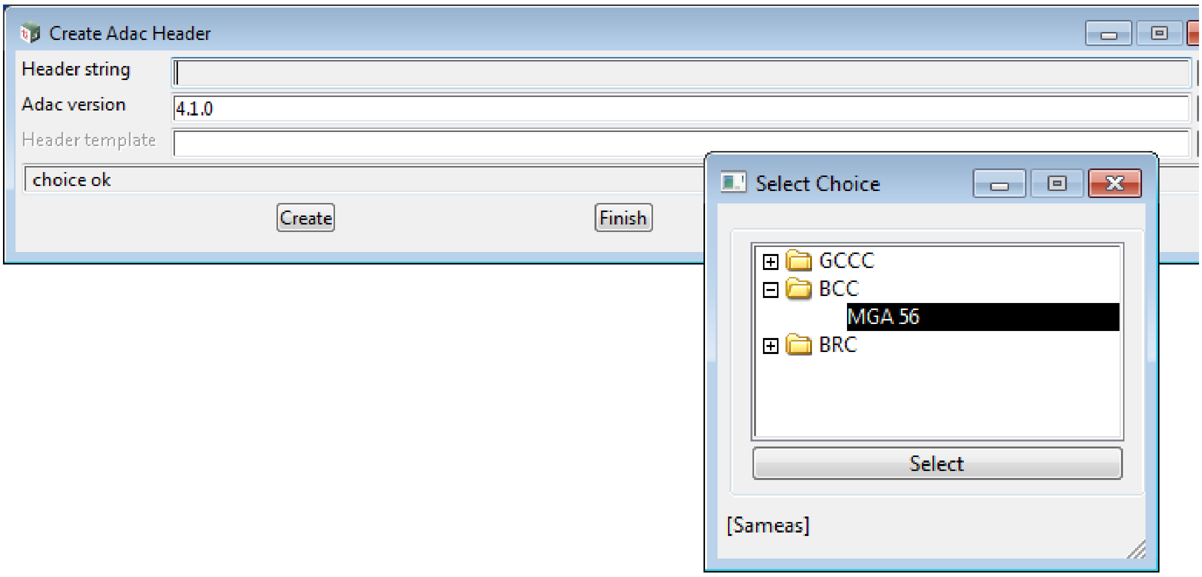
In most firms, many of the values in an *ADAC Header* would be the same for each new ADAC job.

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*.



Create Adac Header

Header string

Adac version 4.1.0

Header template

choice ok

Select Choice

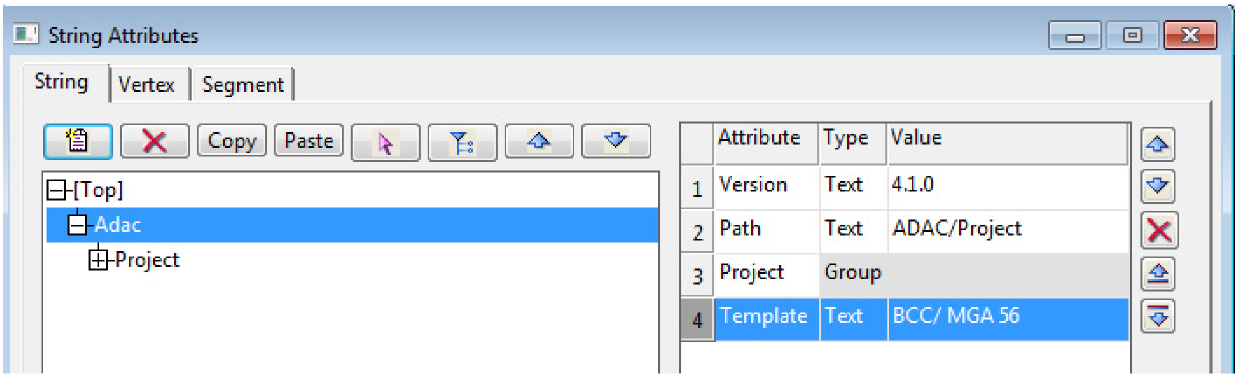
[Sameas]

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***



```
[Top]
  Adac
    Project
```

	Attribute	Type	Value
1	Version	Text	4.1.0
2	Path	Text	ADAC/Project
3	Project	Group	
4	Template	Text	BCC/ MGA 56

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 1.4.9.1.3 Reading in Templates from User_Lib .

Continue to the next section 1.6.4.2 ADAC Asset Templates or return to 1.6.4 Setting Up ADAC Templates .

1.6.4.2 ADAC Asset Templates

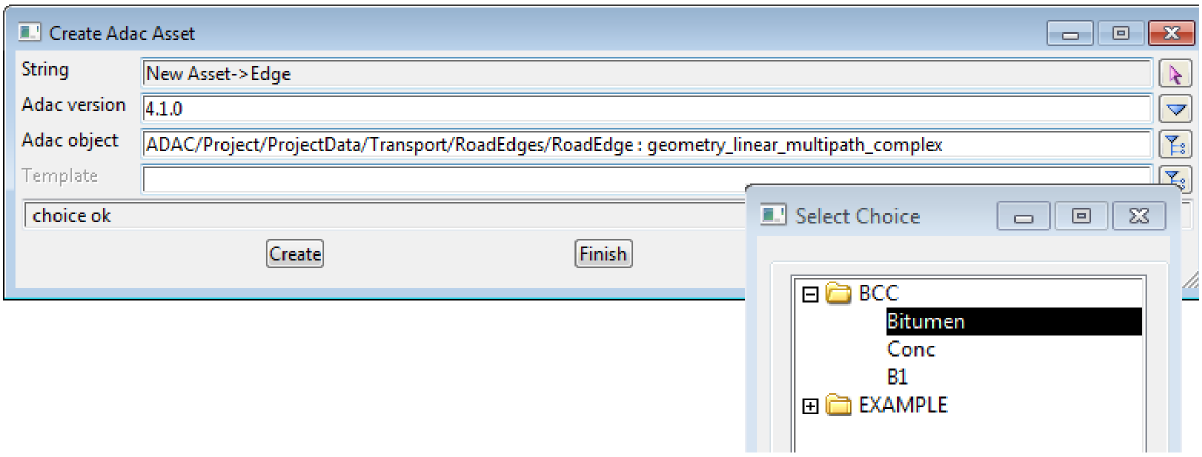
When creating *ADAC Assets* by hand, there would be some the values in an *ADAC Asset* that would be the same each time the *ADAC Asset* was use in ADAC job.

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*.



String: New Asset->Edge
Adac version: 4.1.0
Adac object: ADAC/Project/ProjectData/Transport/RoadEdges/RoadEdge : geometry_linear_multipath_complex
Template:
choice ok

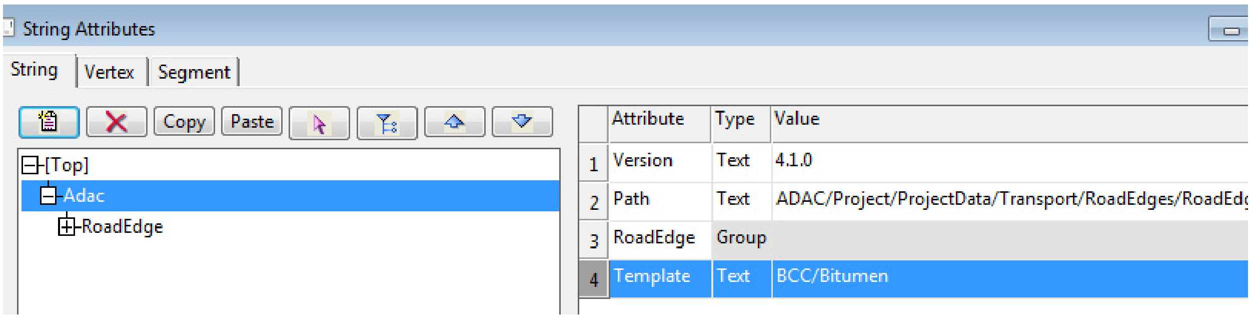
- BCC
  - Bitumen
  - Conc
  - B1
- EXAMPLE

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***



	Attribute	Type	Value
1	Version	Text	4.1.0
2	Path	Text	ADAC/Project/ProjectData/Transport/RoadEdges/RoadEdg
3	RoadEdge	Group	
4	Template	Text	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 1.4.9.1.3 Reading in Templates from User_Lib .

Continue to the next section 1.6.5 Setting Up Your User Keys or return to 1.6 Setting Up for ADAC .

1.6.5 Setting Up Your User Keys

Two thing 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 functions 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 1.6 Setting Up for ADAC .