PoolEdit

User Manual

v1.5.0

Jouko Kalmari
Matti Öhman

2019-11-18

Table of Contents

1 Foreword

PoolEdit is an editor for creating graphical user interfaces (GUIs) for ISO 11783 compliant virtual terminals. The ISO 11783 standard, also known as ISOBUS, defines communication network for tractors and agricultural implements. The PoolEdit editor has been developed in the Automation technology laboratory at Helsinki University of Technology as a part of the Farmix project in 2007.

The editor uses extensible markup language (XML) for saving and editing graphical user interfaces. The collection of graphical user interface objects which makes up the GUI is called an object pool in the standard jargon. The PoolEdit XML format is based on the IsoAgLib format with some minor changes. XML is also used as an export format. The implement control application can read the XML document, generate the ISO 11783 binary presentation and load it to the virtual terminal. Because the XML file is parsed at runtime when the VT properties are known, a smart parser can resize the GUI objects and reduce color depth accordingly. Another benefit of the XML format is that it allows the use of symbolic names for referencing the GUI objects where as the binary format relies solely on numeric object IDs.

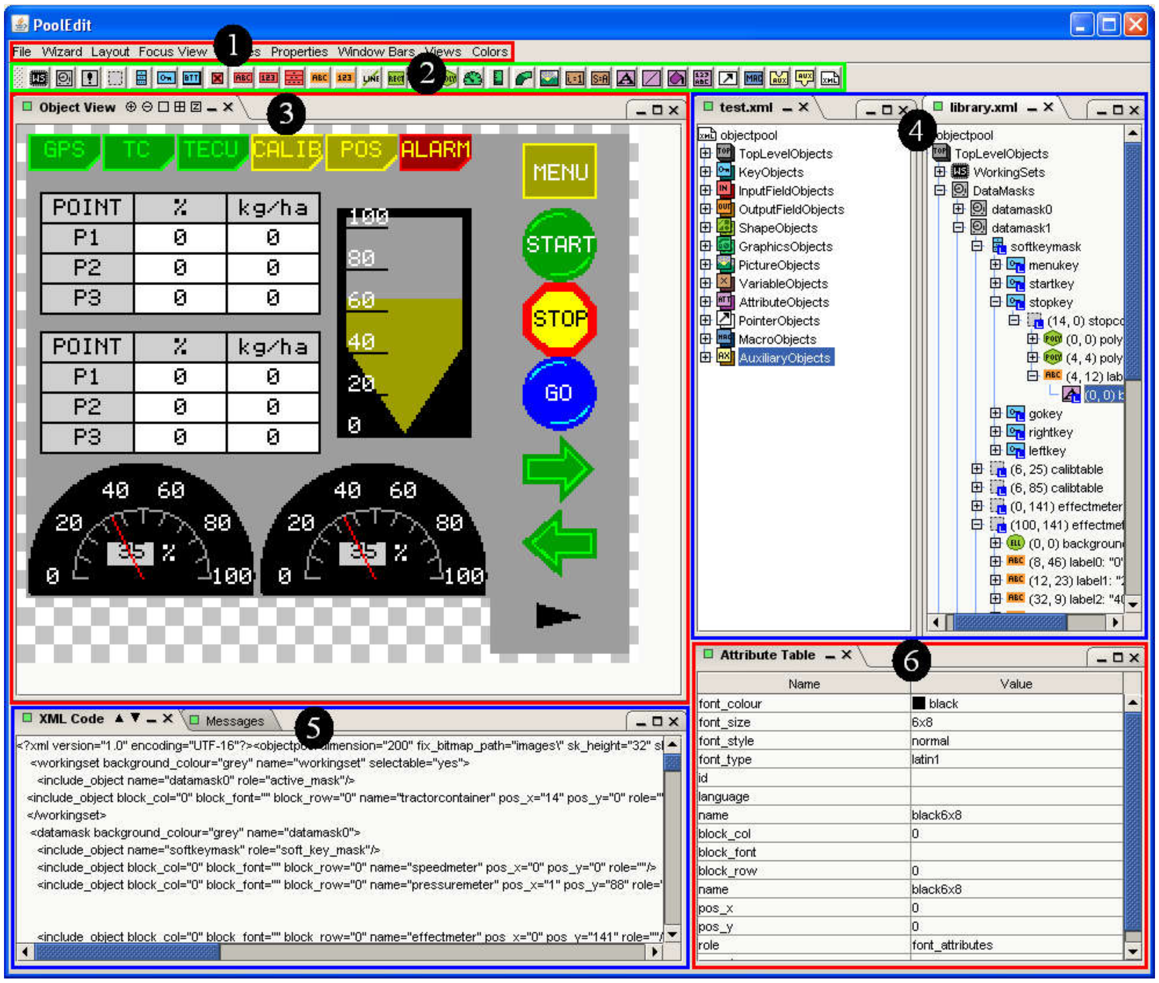
Example XML-code generated by PoolEdit

```
<container height="53" name="pressuremeter" width="64">
  <ellipse ellipse_type="closedsection" end_angle="220" height="64"
  name="background" pos_x="0" pos_y="0" start_angle="320" width="64">
    <include_object name="black" role="fill_attributes"/>
    <include_object name="grey1" role="line_attributes"/>
  </ellipse>
  <outputstring background_colour="white" height="8"
  horizontal_justification="middle" name="label0" options="transparent"
  pos_x="9" pos_y="42" value="0" width="6">
    <include_object name="white6x8" role="font_attributes"/>
  </outputstring>
</container>
```

PoolEdit is written in the Java programming language, and it has been tested on both Windows and Linux operating systems.

The source code was originally published on the university’s own website. Unfortunately that website was not maintained for years and the server was finally taken offline in 2019. This provided a good opportunity to find a new home for PoolEdit. While moving the project to GitHub it became apparent that the source code could use some maintenance: changes to the Java platform had broken some features and some features had probably never worked quite right. There is also one completely new feature: a separate program was written years ago to import object pools in various formats but it was never published. It is now integrated to PoolEdit as the Import feature. After many small fixes and improvements the version number has been increased from 1.4 to 1.5.0 and the code is again available to a wider audience.

2 Editor Layout



```
<?xml version="1.0" encoding="UTF-16"?><objectpool dimension="200" fix_bitmap_path="images\" sk_height="32" sl
  <workingset background_colour="grey" name="workingset" selectable="yes">
    <include_object name="datamask0" role="active_mask"/>
  <include_object block_col="0" block_font="" block_row="0" name="tractorcontainer" pos_x="14" pos_y="0" role="
  </workingset>
  <datamask background_colour="grey" name="datamask0">
    <include_object name="softkeymask" role="soft_key_mask"/>
    <include_object block_col="0" block_font="" block_row="0" name="speedmeter" pos_x="0" pos_y="0" role=""/>
    <include_object block_col="0" block_font="" block_row="0" name="pressuremeter" pos_x="1" pos_y="88" role="


    <include_object block_col="0" block_font="" block_row="0" name="effectmeter" pos_x="0" pos_y="141" role=""/
```

Name	Value
font_colour	■ black
font_size	6x8
font_style	normal
font_type	latin1
id	
language	
name	black6x8
block_col	0
block_font	
block_row	0
name	black6x8
pos_x	0
pos_y	0
role	font_attributes

PoolEdit's basic layout.

PoolEdit has different views and some other components for managing and editing the object pools:

1. Menus

Menus are used for issuing file operations, running wizards and adjusting various settings.

2. Objects

New objects are created by dragging them from the object toolbar to the tree model.

3. Object View

The object view shows how the objects will be rendered on the virtual terminal's display. The object view can be zoomed and scrolled and it has various options for precise placement of objects. It is also possible to delete, resize and move objects in this view.

4. Tree View

The tree view shows the objects in a single object pool. Although object pools are inherently DAGs[1], they can be visualized as trees. The tree view shows the types of objects, their names and how those objects are related. Multiple object pools can be open at the same time. Objects can be easily copied between and within the trees.

5. XML and Message Views

The XML view can be used for directly editing the generated XML code. This is useful for debugging but not needed for normal use. The message view (not visible in the image) is for displaying warning and other messages.

6. Attribute Table

The attribute table is shows the attributes and the attribute values of the selected object. The table has also different editors for editing different attribute types. For example, the GUI designer can select a colour by picking it from a colour list. Object attributes include things like object's width and colour while link attributes specify the position of the object or the role it plays in its parent object.

[1] Directed, acyclic graphs.

3 Structure of an Object Pool

An object pool is a collection of objects. Some objects may have other objects inside them. PoolEdit has two ways of setting objects inside each other. An object can be directly inserted as a child into its parent object. Alternatively, a special link object can be inserted that points to the actual child object indirectly.

3.1 Root Object

The root element of the XML document is called objectpool. The objectpool element has the following attributes:

- The *dimension* attribute defines the width and height of data and alarm masks. The ISO 11783-6 standard specifies the minimum resolution of 200x200 pixels. Using this resolution allows the designs to be scaled upwards in the control application before they are loaded to the virtual terminal.
- The *sk_width* and *sk_height* attributes define the width and height of designator areas, respectively. The standard specifies minimum designator size of 32x60 pixels. Using this resolution allows the designs to be scaled upwards.
- The *fix_bitmap_path* and *std_bitmap_path* attributes define paths to image files. The *fix_bitmap_path* is mainly for compatibility with the IsoAgLib format and older virtual terminals.

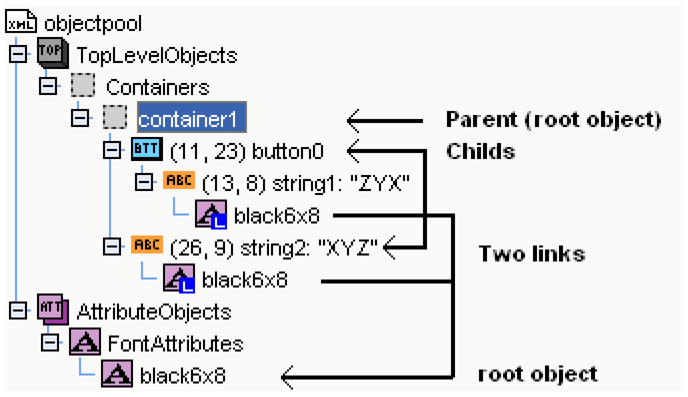
An example root element is show below:

```
<objectpool  dimension="200"  fix_bitmap_path="images\"  sk_height="32"
    sk_width="60" std_bitmap_path="images\">

    ...

</objectpool>
```

3.2 Linking

Linking makes it possible for a single object to have many parent objects. A link can be distinguished from a letter ‘L’ in its logo in the Tree view. Links are allowed only inside other objects. No standalone link objects may exist. The linked objects must be located at the root level of the pool. It is not possible to make links that point any deeper in the tree.

Usually it is a good idea to use attribute objects with links, so that many identical objects are not created (see chapter 4.1.2 for automatically combining identical objects). Linking is based on the object names. Therefore there cannot be two objects with the same name at the root level.



Simplified Tree View of a pool (unnecessary category objects are not shown)

Picture above shows a pool that has container and font attribute objects in the root level. The container has two child objects in it; a button and an output string. The button has also a string as a child. Both output strings link to the same font attributes object.

3.3 Object Types

The ISO 11783 standard specifies a set of object types. The object types are denoted by unique icons in the Tree view. The icons of similar object types have similar colors, which makes them easier to recognize.

Icons for different objects

WS	Working set	ABC	Output string	S=A	String variable
	Data mask	123	Output number	A	Font attributes
!	Alarm mask	LINE	Line		Line attributes
	Container	RECT	Rectangle		Fill attributes
	Soft key mask	ELL	Ellipse	123 ABC	Input attributes
	Soft key	POLY	Polygon	↗	Object pointer
BTT	Button		Meter	MAC	Macro
☒	Input boolean		Linear bar graph	AUX	Auxiliary function
ABC	Input string		Arched bar graph	AUX	Auxiliary input
123	Input number		Picture graphic		
	Input list	L=1	Number variable		

Objects in the root level are grouped by the type (e.g. all containers are under Containers node) and these are set under category objects (e.g. WorkingSets, DataMasks, AlarmMasks and Containers are under TopLevelObjects). These category objects are not real objects; they are there just for grouping objects.

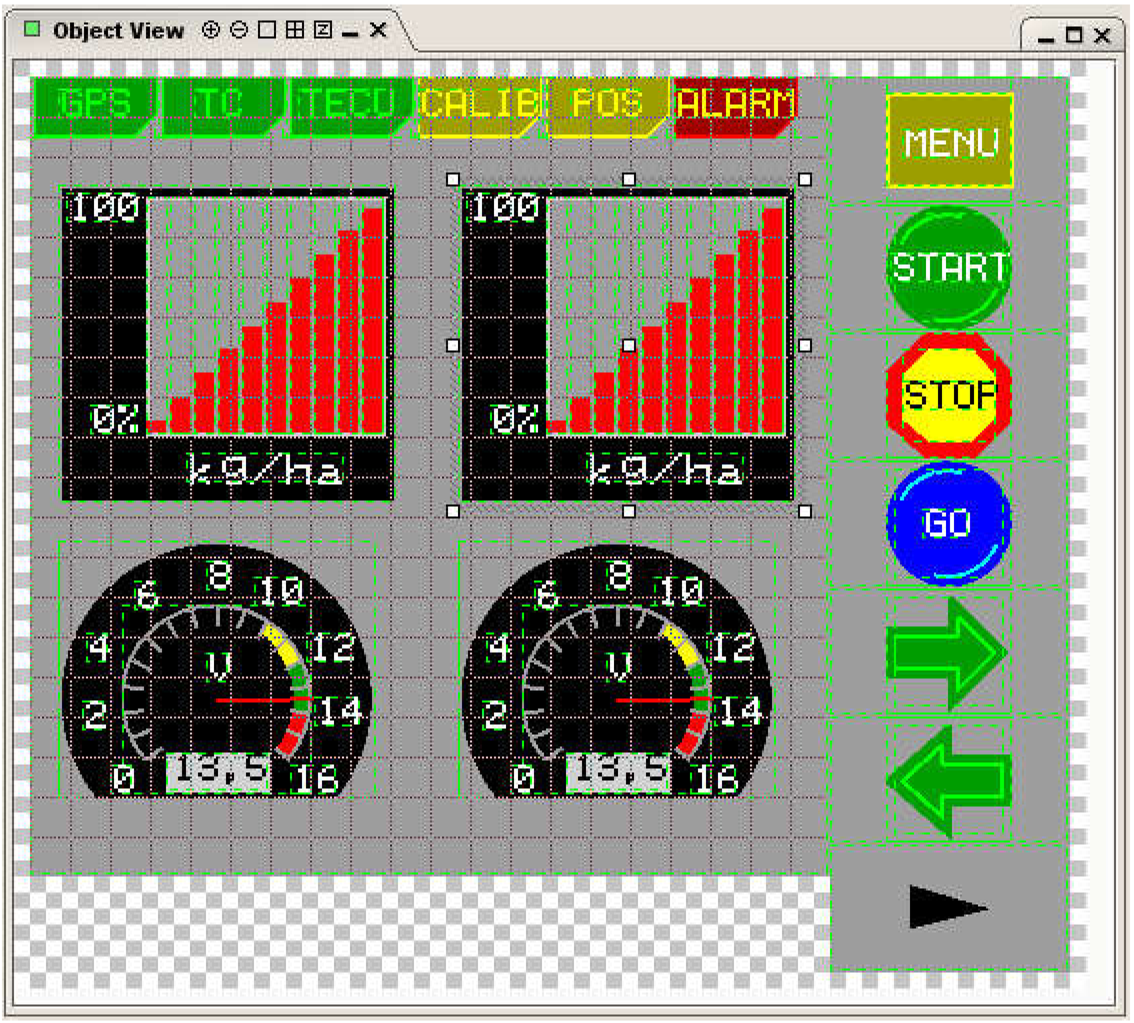
The Tree view shows also a couple of object types that are not real ISO 11783 objects. Polygon objects have three or more Point objects (●) as children. The Point objects are used to denote corners of the polygon. The Working set object can have Language code objects (LNG), which specify the supported languages. Command objects (C) are used within Macro objects for describing command sequences. However, there are no special event objects. Instead, events are specified by the role attribute of the Macro object.

4 Editing

4.1 Object View

The Object view shows how the object pool will look on a virtual terminal. An object can be selected by clicking it with a mouse. If there are multiple objects on top of each other, the topmost one will be selected. If it is not possible to select the desired object by using the Object view, use the Tree view instead.

The selected object is surrounded by white squares. The object can be resized and moved by dragging these boxes. It is possible to move objects so that they have negative coordinates, but as the ISO 11783 standard does not currently allow negative coordinates, they should be avoided.



Object view example. Pool is zoomed, grid is on, borderlines are on, tiling is off and one object is selected.

4.1.1 Object View Buttons

There are five buttons in the upper left corner of the Object view that change the visualization of the object pool.

⊕/⊖ Zoom in / zoom out (zooming can be also done with mouse wheel).

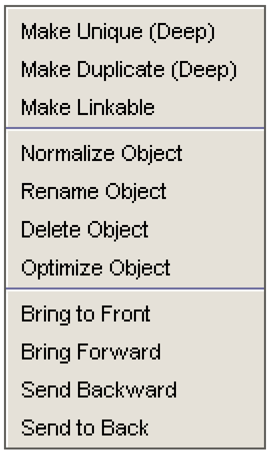
□ Show / hide borders. Borders are a thin green dotted line

⊞ Show / hide grid. The grid is always drawn on the parent of the selected object.

$\boxed{Z}$ Show / hide tiling. Tiling makes zoomed objects look like they would on the virtual terminal, when no scaling is done (the pixels are magnified).

4.1.2 Drop Menu

Right clicking an object in the Object view or Tree view opens a drop menu.



Make Unique

Creates a unique object from the link. Further changes to this object do not affect the original object. The selected object must be a link.

Make Duplicate

Creates a copy of the object. Objects can also be copied by dragging and dropping.

Make Linkable

Copies the object to the root level and replaces the original with a link.

Normalize Object

Resizes the object so it is as small as possible. This is great for resizing string and number fields but it also works for many other object types.

Rename Object

Renames the object and all links pointing to it.

Delete Object

Deletes the object. The 'Del' button works as well.

Optimize Object

Recursive search that tries to find objects that could be replaced with links. Objects that are replaced must be identical. This is great for replacing identical attribute objects. Remember that objects can be referenced only if they are located at the root level!

Bring To Front

Moves the object to the front. It will be drawn last.

Bring Forward

Moves the object one step forward.

Send Backward

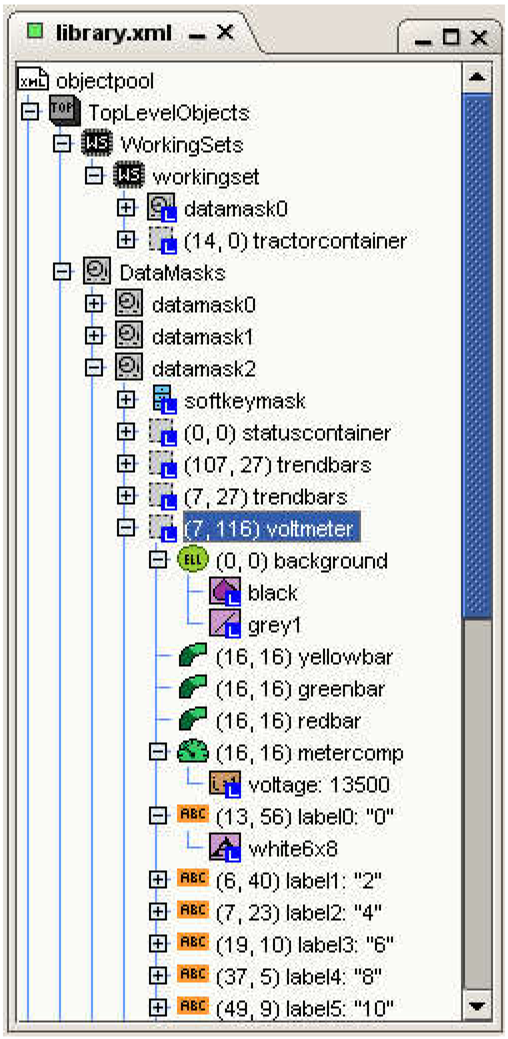
Moves the object one step backward.

Send to Back

Moves the object to the back. It will be drawn first.

4.2 Tree View

Tree view shows the structure of the object pool as a tree structure. Objects that are inside other objects are shown as children. Links are marked with a letter 'L' over the object type icon.



```
library.xml
objectpool
⊟ TopLevelObjects
   ⊟ WorkingSets
      ⊟ workingset
         ⊞ datamask0
         ⊞ (14, 0) tractorcontainer
   ⊟ DataMasks
      ⊞ datamask0
      ⊞ datamask1
      ⊟ datamask2
         ⊞ softkeymask
         ⊞ (0, 0) statuscontainer
         ⊞ (107, 27) trendbars
         ⊞ (7, 27) trendbars
         ⊟ (7, 116) voltmeter
            ⊟ (0, 0) background
                  black
                  grey1
              (16, 16) yellowbar
              (16, 16) greenbar
              (16, 16) redbar
            ⊟ (16, 16) metercomp
                  voltage: 13500
            ⊟ (13, 56) label0: "0"
                  white6x8
            ⊞ (6, 40) label1: "2"
            ⊞ (7, 23) label2: "4"
            ⊞ (19, 10) label3: "6"
            ⊞ (37, 5) label4: "8"
            ⊞ (49, 9) label5: "10"
```

4.2.1 Drag and Drop

The easiest way to copy objects and to create links is by dragging and dropping with a mouse. Objects can be copied within and between documents.

Copy

Copying is done by dropping the selected object over its new parent object. Alternatively, the selected object can be dropped between its new sibling objects.

Copy is implemented as a deep copy which means that all children are copied as well and that all links are replaced by corresponding actual objects.

Link

A link is done similarly as copying, except that ctrl-shift must be pressed throughout the operation.

Move

Not currently supported as moving objects between documents can be dangerous!

4.2.2 Drop Menu

Right clicking an object in the Tree view opens a drop menu that is identical to the drop menu in the Object view. See chapter 4.1.2.

4.2.3 Attribute and Variable Objects

The IsoAgLib XML format denotes attribute and variable references differently than other references. Attribute and variable references are specified by XML attributes, while other references are described by XML elements:

```
<workingset active_mask="startMask">
   <include_object name="logo" pos_x="16" pos_y="6"/>
</workingset>
```

This is rather unfortunate as while the included 'logo' reference could be replaced by the actual object, the included 'startMask' reference could not.

The PoolEdit XML format takes a more systematic approach:

```
<workingset>
    <include_object name="startMask" role="active_mask"/>
    <include_object name="logo" pos_x="16" pos_y="6"/>
</workingset>
```

All references are denoted by 'include_object' elements. The same way as the graphical object specifies its position relative to the parent object, the non-graphical object specifies its role in the parent object. For example, when a Font attributes object is used it should have a Role attribute set to 'fontattributes'. The role attributes are set automatically to their default values by PoolEdit. If a non-default value is desired, the user can change the value from the Attribute table, which lists all possible values for the selected object.

4.2.4 Macros

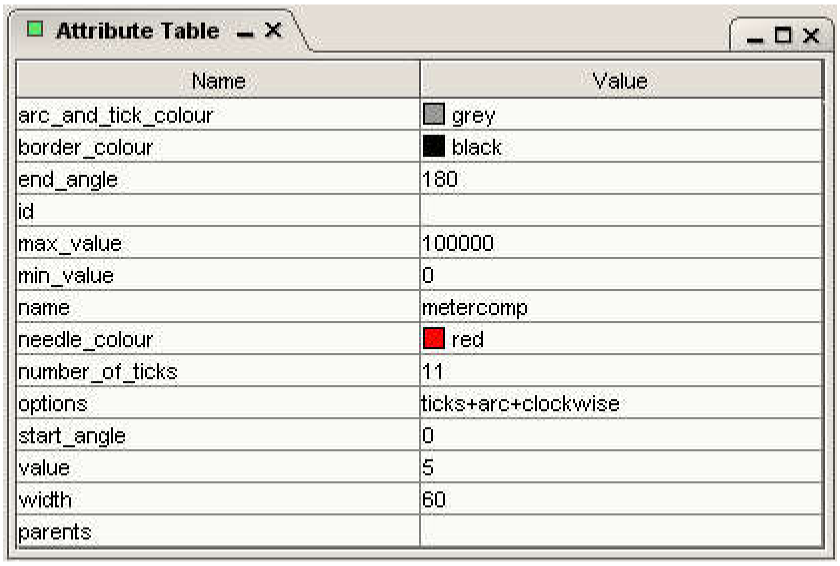
Macros are command sequences that can be triggered by various events. The triggering event is specified by the Role attribute when a macro is used in an object.

```
<macro name="changeMask" role="on_key_press">
   <command_change_active_mask child_id="600" parent_id="1"/>
</macro>
```

Currently, the user has to use object ID in the command objects and fix the appropriate object IDs. In a future release, we are planning to use symbolic names instead of object IDs. However, there are some implications that have to be resolved first: other objects cannot reference their parents (as it would create loops) but command objects typically do (and it should not be considered a loop). Also symbolic names can only point to the root level (at least for now) but object IDs can point to anywhere (but they have to be globally unique).

4.3 Attribute Table

The Attribute table shows the attributes and the attribute values of the selected object. Most of attributes map directly to the ISO 11783 standard. Some attributes are missing as they are replaced by the Role attribute in a child object. There are also some attributes that are specific to the PoolEdit XML format.



Name	Value
arc_and_tick_colour	⬜ grey
border_colour	⬛ black
end_angle	180
id	
max_value	100000
min_value	0
name	metercomp
needle_colour	🟥 red
number_of_ticks	11
options	ticks+arc+clockwise
start_angle	0
value	5
width	60
parents	

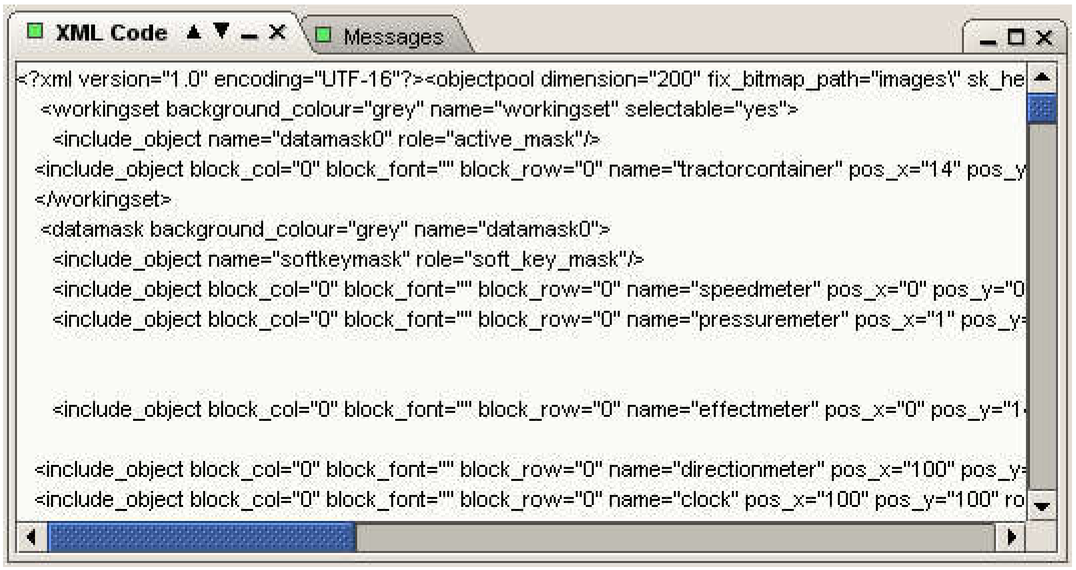
Attribute table example. The table shows the attributes of the selected Output meter object.

If a fixed object ID is needed for a certain object, it can be set. Normally IDs are created when the object pool is exported as an embedded XML file.

When a link is selected, the attribute table shows both the object attributes (name, color etc.) and the link attributes (name, position etc).

4.4 XML View

The XML View can be used to examine and edit the generated XML code. The code in the editor is not automatically updated when pool is changed or vice versa. Before editing the code it has to be manually updated by pressing the down arrow (▼) and when editing is done pressing the up arrow (▲) loads the XML code back to the object pool.



```
<?xml version="1.0" encoding="UTF-16"?><objectpool dimension="200" fix_bitmap_path="images\" sk_he
  <workingset background_colour="grey" name="workingset" selectable="yes">
    <include_object name="datamask0" role="active_mask"/>
 <include_object block_col="0" block_font="" block_row="0" name="tractorcontainer" pos_x="14" pos_y
 </workingset>
  <datamask background_colour="grey" name="datamask0">
    <include_object name="softkeymask" role="soft_key_mask"/>
    <include_object block_col="0" block_font="" block_row="0" name="speedmeter" pos_x="0" pos_y="0
    <include_object block_col="0" block_font="" block_row="0" name="pressuremeter" pos_x="1" pos_y=


    <include_object block_col="0" block_font="" block_row="0" name="effectmeter" pos_x="0" pos_y="1

 <include_object block_col="0" block_font="" block_row="0" name="directionmeter" pos_x="100" pos_y=
 <include_object block_col="0" block_font="" block_row="0" name="clock" pos_x="100" pos_y="100" ro
```

XML View Example.

5 Creating, Loading, Saving and Exporting Documents

5.1 Creating New Object Pool

The *New* menu option creates a new object pool by loading a document template.

5.2 Loading and Saving

The *Load* menu option loads an existing document. Loading of both PoolEdit and IsoAgLib XML files are supported.

The *Save* and *Save as…* menu options use the PoolEdit XML format to save the active document. When saving a document make sure you have selected the appropriate document to save!

5.3 XML Formats

There are three types of XML formats used by PoolEdit.

Format Name	**Loads**	**Saves**	**Exports**	**Usage**
PoolEdit XML	x	x		Native format
IsoAgLib XML	x *)		x *)	Legacy IsoAgLib format
Embedded XML			x	Self-contained export format

*) not well tested

5.3.1 IsoAgLib XML

This is the original IsoAgLib XML format on which the other formats are based on. We have not used the actual IsoAbLib library and have not done any compatibility testing. There are likely to be so minor issues but in principle the conversions should work.

5.3.2 PoolEdit XML

The native XML format of PoolEdit resembles IsoAglib's format. Biggest difference is that 'include object' elements are used to link attributes, variables, data masks and soft key masks to objects while the IsoAglib format uses XML attributes to do the same job. In PoolEdit XML links can only point to objects on the root level.

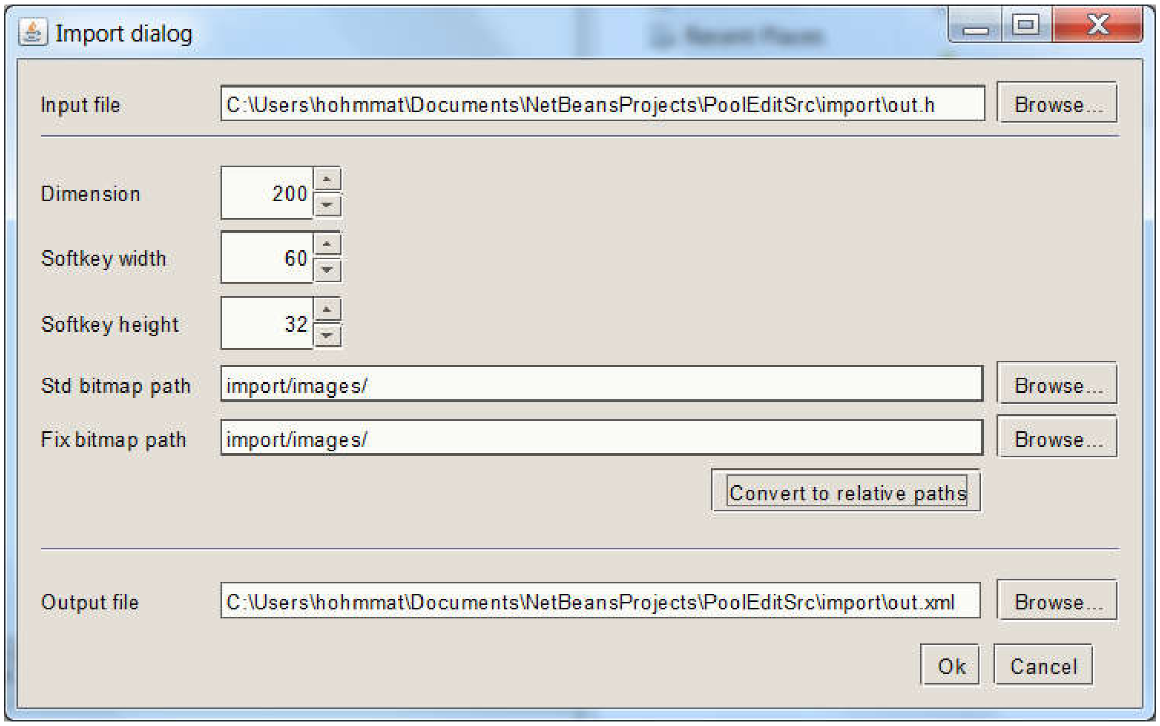
5.3.3 Embedded XML

Embedded XML is designed so that it can be transformed to ISO 11783 binary format and send to a virtual terminal. It is derived from the PoolEdit XML format, but there few differences. The embedded XML file has picture data embedded to it by using base64 encoding. This removes the need for separate picture files. In addition, all objects have unique IDs. This makes it possible to detect ID conflicts earlier and makes the subsequent processing easier.

In embedded XML files, all objects are marked as mask objects or designator objects. Mask objects are displayed on either data or alarm masks. Designator objects can appear on working set logo, soft key masks, or auxiliary function and input designators. This classification is done to make object scaling easier in the case different scale factors have to be used for mask and designator objects. If an object is used on both mask areas and designator areas, its use attribute is set to value "both", which is basically an error condition. Do not use the same objects on both mask and designator areas!

6 Importing object pools

The *Import* menu option opens the Import dialog window:


Input file: C:\Users\hohmmat\Documents\NetBeansProjects\PoolEditSrc\import\out.h
Dimension: 200
Softkey width: 60
Softkey height: 32
Std bitmap path: import/images/
Fix bitmap path: import/images/
Output file: C:\Users\hohmmat\Documents\NetBeansProjects\PoolEditSrc\import\out.xml

The Import dialog window.

The input file field contains the full path to the file being imported. Currently 3 formats are supported:

- **.sav* files (PoolEdit's predecessor used this as the export format)
- **.iop* files (ISOBUS Object Pool file, binary data file as defined in the standard)
- **.h* files generated by the PoolEdit parser

The *dimension, sk_width* and *sk_height* attributes can be set in the dialog window or they can be set later by editing root node's attributes in the attribute editor. Likewise the *std_bitmap_path* and *fix_bitmap_path* are attributes of the root node. However, it is important to set the *std_bitmap_path* attribute now because it defines the directory where the image files will be created!

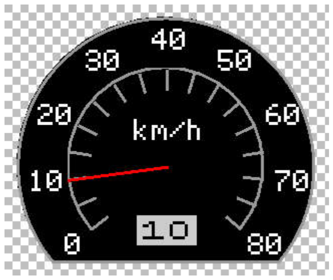
The output file must have XML file extension!

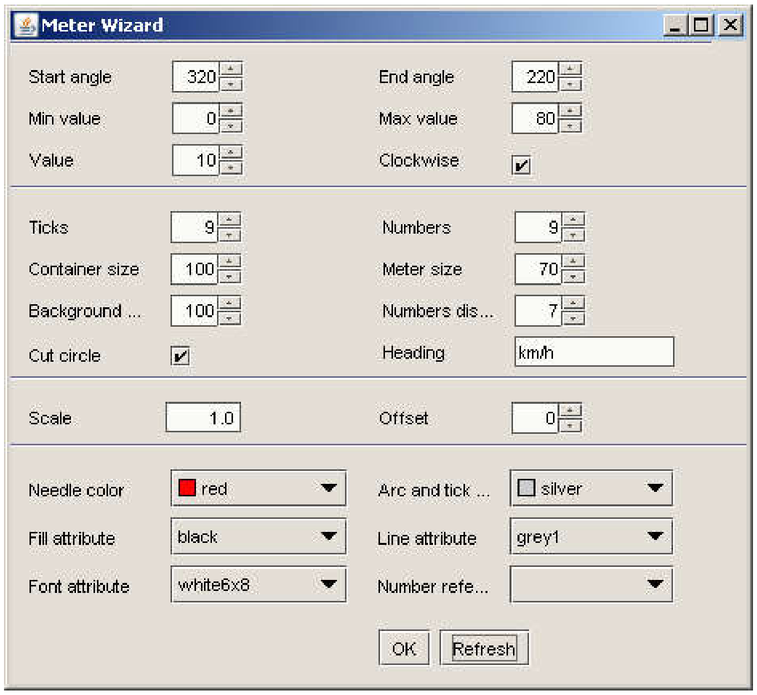
7 Wizards

The PoolEdit wizards are little, prewritten programs which generate XML code according to the user's specification. All wizards have a dialog screen for collecting user input. The user can preview the result by pressing the 'Refresh' button, which will generate a new composite object under the selected object. To actually see the object, the user has to navigate to the new object in the Tree view. When the user is sufficiently pleased with the result, pressing the 'Ok' button will close the wizard. When the wizard is closed, the user can fine tune the created objects as if they were created manually.

7.1 *Meter Wizard*

The Meter wizard is used to create fancy meters with numeric scales and solid backgrounds.

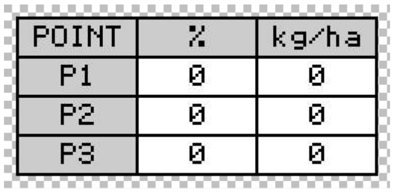




Start angle	320	End angle	220
Min value	0	Max value	80
Value	10	Clockwise	✓
Ticks	9	Numbers	9
Container size	100	Meter size	70
Background ...	100	Numbers dis...	7
Cut circle	✓	Heading	km/h
Scale	1.0	Offset	0
Needle color	🟥 red	Arc and tick ...	☐ silver
Fill attribute	black	Line attribute	grey1
Font attribute	white6x8	Number refe...	

7.2 Table Wizard

The Table wizard is used to create tables with multiple rows and columns.



POINT	%	kg/ha
P1	0	0
P2	0	0
P3	0	0

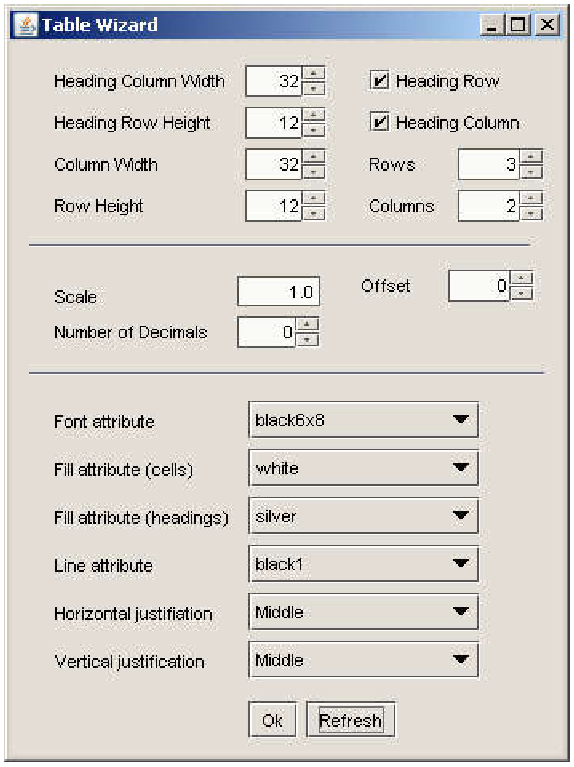


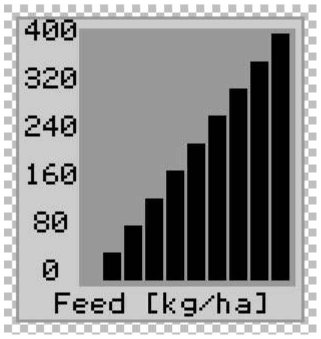
Table Wizard

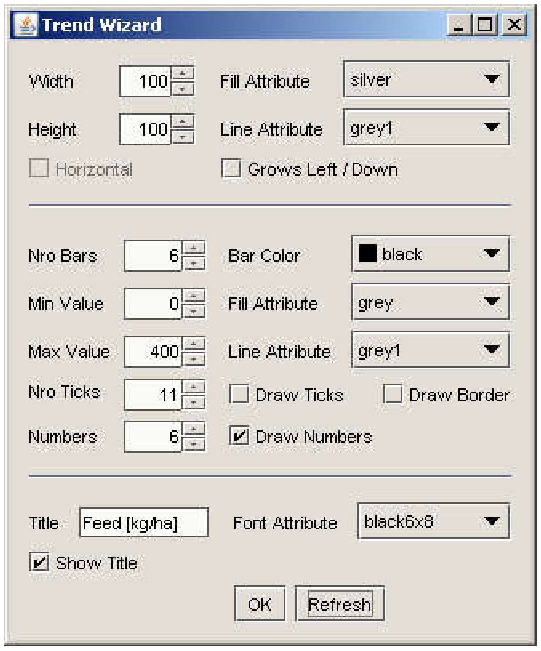
Heading Column Width	32	✓ Heading Row	
Heading Row Height	12	✓ Heading Column	
Column Width	32	Rows	3
Row Height	12	Columns	2
Scale	1.0	Offset	0
Number of Decimals	0		

Font attribute	black6x8
Fill attribute (cells)	white
Fill attribute (headings)	silver
Line attribute	black1
Horizontal justifiation	Middle
Vertical justification	Middle

7.3 Trend Wizard

The Trend wizard is used to create trend display component out of basic linear bar graph components.



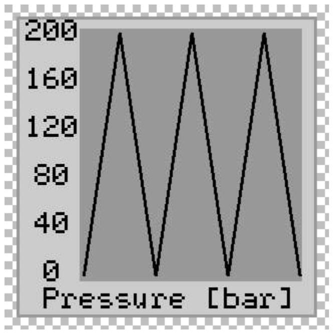


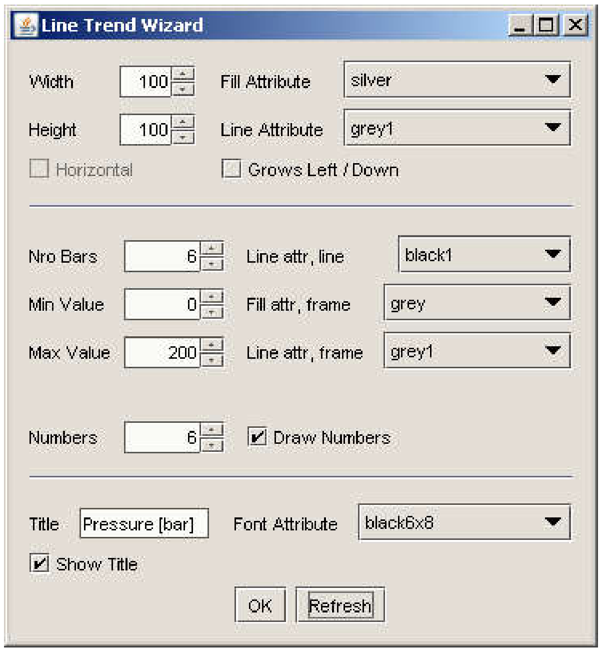
Trend Wizard

Width	100	Fill Attribute	silver
Height	100	Line Attribute	grey1
☐ Horizontal		☐ Grows Left / Down	
Nro Bars	6	Bar Color	■ black
Min Value	0	Fill Attribute	grey
Max Value	400	Line Attribute	grey1
Nro Ticks	11	☐ Draw Ticks	☐ Draw Border
Numbers	6	✓ Draw Numbers	
Title	Feed [kg/ha]	Font Attribute	black6x8
✓ Show Title			

7.4 Line Trend Wizard

The Line trend wizard is quite similar to the Trend wizard except it uses lines instead of linear bar graphs.





Width	100	Fill Attribute	silver
Height	100	Line Attribute	grey1
☐ Horizontal		☐ Grows Left / Down	
Nro Bars	6	Line attr, line	black1
Min Value	0	Fill attr, frame	grey
Max Value	200	Line attr, frame	grey1
Numbers	6	☑ Draw Numbers	
Title	Pressure [bar]	Font Attribute	black6x8
☑ Show Title			

8 Planned Features

- Undo / Redo
- Performance improvements (especially the Table wizard is very slow when creating large tables)
- In place editing for input and output string objects
- Navigation macro wizard (for creating navigation macros for soft keys)
- Preview function for showing the effects of scaling and color reduction on virtual terminal's display
- Better macro / command support
- Hierarchical name space (compare with Unix style file system with links, absolute and relative paths)
- Better validation of object pools (many more things should be checked)
- Better error handling
- Changeable grid spacing and snap to grid features (we have the basic grid already)
- Better component libraries and better ways to organize them