# DOORS

## Plugin Manual for ASCE 4.2

- File name: adelard doors object import.xml
- Version: 1.0.0
- Applies to: All networks

Contents

## 1 Description

A DNR plugin for inserting content from IBM DOORS (Classic) databases[1]. This plugin lets you insert traceable content that is managed within DOORS.

This plugin provides two DNRs which may be accessed via the Insert-DNR menu within the ASCE node editor.

## 2 IBM Rational DOORS

IBM Rational DOORS[2] is a requirements management tool, typically used to manage functional and safety requirements on system engineering projects. Projects may also use DOORS to manage other project data, such as test cases.

The DOORS system provides a user interface to create a number of *Formal Modules* within a *Project* and *Folder* structure (see Figure 1). Each module contains a number of objects (see Figure 2). These objects typically represent the requirements being managed for a system under consideration.

---

[1] This plugin is not compatible with the web based "DOORS Next Generation" system

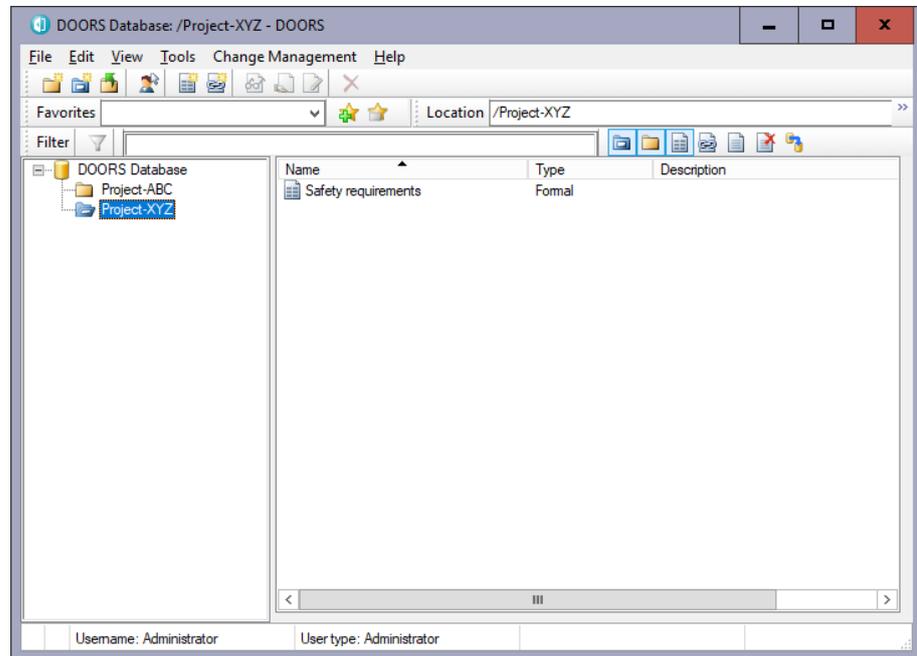[2] See https://www.ibm.com/us-en/marketplace/rational-doors
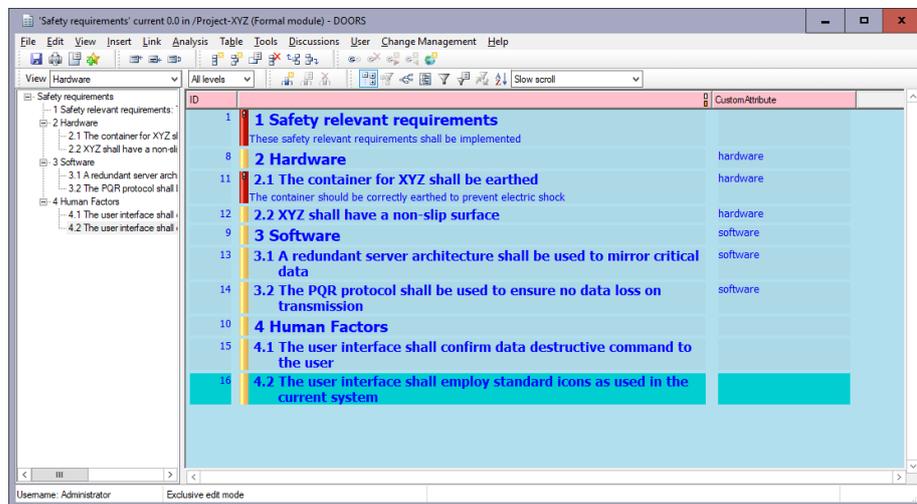
Figure 1: DOORS Database containing modules



Figure 2: A formal module in DOORS containing safety requirements

This user guide assumes you are familiar with using DOORS to create and manage requirements.

The DOORS plugin for ASCE will reference the module according to its full path, which is composed of the project or folder structure together with the module name. From the example above this would be:

• Folder name/path to module: /Project-XYZ
• Module name: Safety requirements

Hence the full path is /Project-XYZ/Safety requirements

## 3  ASCE Plugin

This plugin provides two DNRs which summarise content held in a DOORS database:

1.  a specific object in a module – see Section 3.1
2.  all objects in a particular module – see Section 3.2

To insert each DNR choose the appropriate menu option from the Insert-DNR menu in the ASCE node editor (see Figure 3).
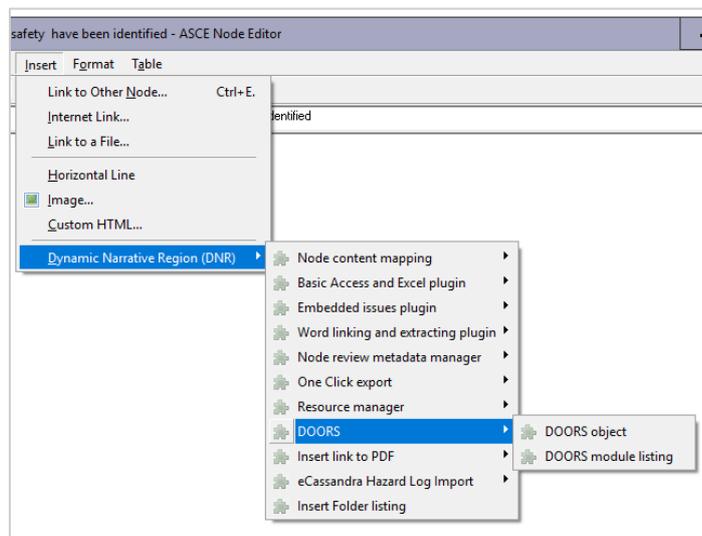


**Figure 3: Menu for the DOORS plugin DNRs within the ASCE node editor.**

The DOORS plugin for ASCE connects to the DOORS application, so you must be logged into DOORS for the integration to be able to access the data.

### 3.1  DOORS Object DNR

This DNR summarises the content of a specific object within a formal module. It could be used for example to track the status of an important requirement within DOORS about which a claim will need to be made in the safety case.

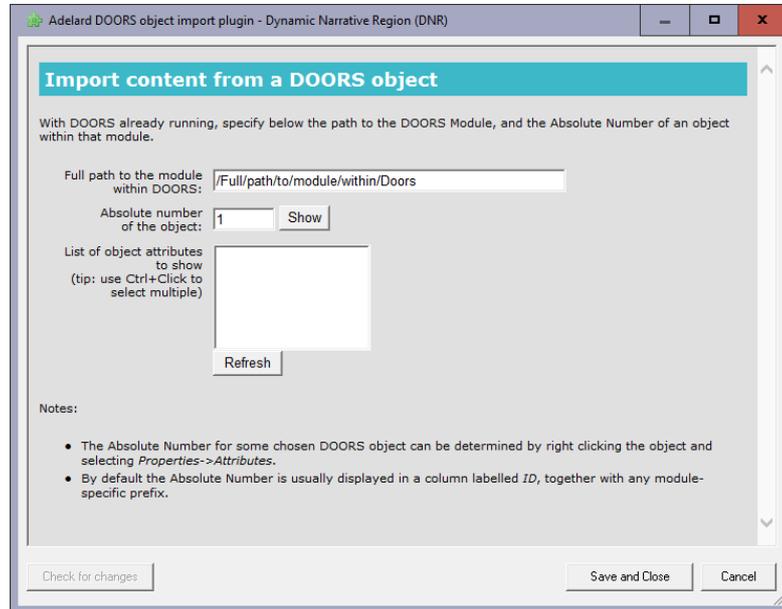Insert the DNR then click on it to open its editor window (see Figure 4).

Figure 4: DNR Editor for DOORS object

Enter the full path to the DOORS module, which for our example is /Project-XYZ/Safety requirements then press the [Refresh] button to list the available attributes – see Figure 5.
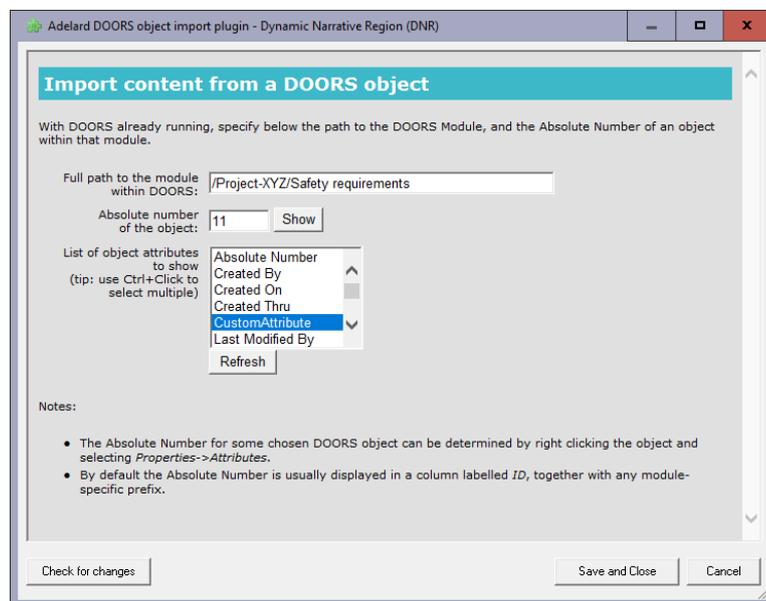


Figure 5: DNR Editor for DOORS object, showing attributes

Choose any attributes to be displayed, then press [Save and Close] to have the content of the object rendered in the node narrative – see Figure 6
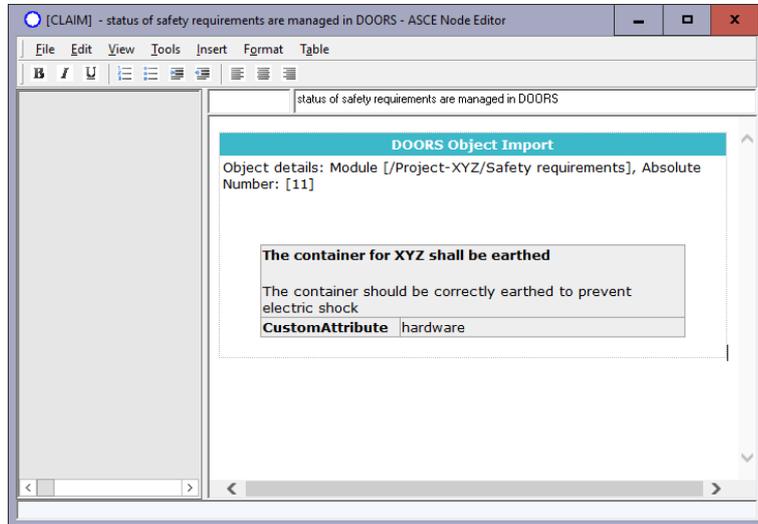
**Figure 6: DNR rendered of a single object within the node narrative**

## 3.2    DOORS Module listing DNR

You can insert a listing of all objects in a module, optionally filtered.

From the Insert-DNR menu in ASCE node editor choose the *DOORS Module listing* DNR. Click on the DNR to open its editor – see Figure 7.
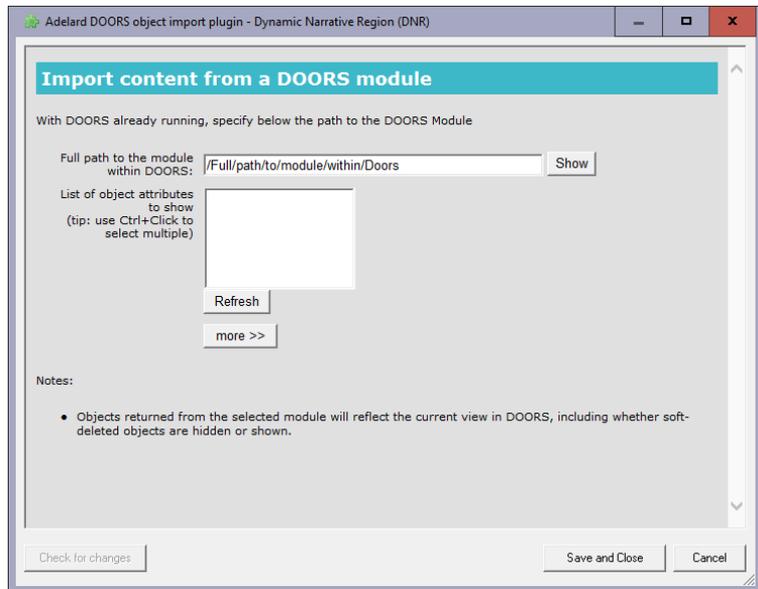


**Figure 7: DNR editor for DOORS module listing**

Enter the full path to the DOORS module, which for our example is /Project-XYZ/Safety requirements then press the [Refresh] button to list the available attributes – see Figure 8.
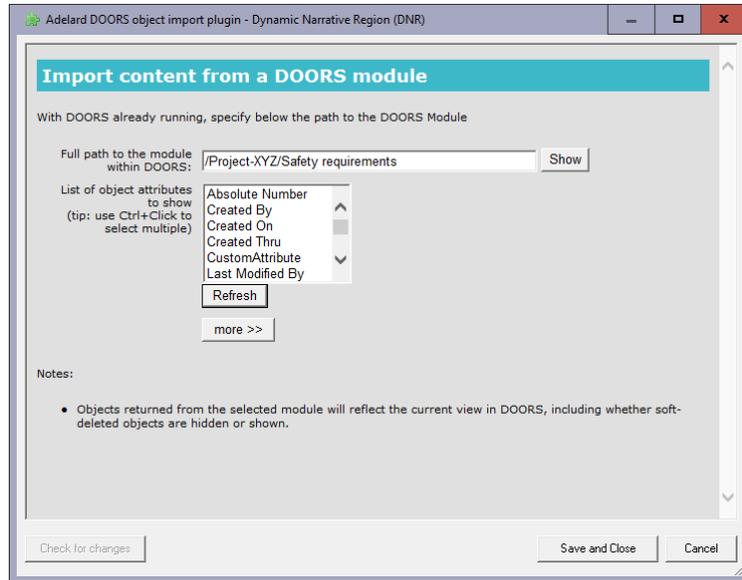
Figure 8: DNR editor showing module attributes

From the list of attributes are shown, you can choose any specific attributes to be shown. Then press [Save and Close] to render the DNR into the node narrative content Figure 9.
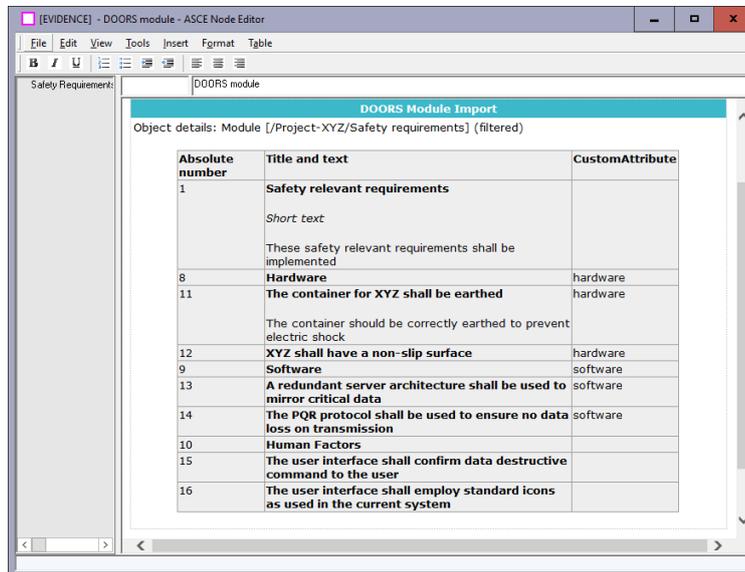


Figure 9: Rendered DNR showing content from a specified DOORS module.

The objects in the specified module will be returned in a table, with the following columns:

• Absolute Number
• Object Heading, Object short text, Object text
• <any specific attributes chosen>

Objects returned from the selected module will reflect the current view in DOORS, including whether soft-deleted objects are hidden or shown.

### 3.2.1   Filtering the results using XPath

An advanced feature of the module listing DNR is that you can filter the results shown. To define a filter, expand the [more] button and edit the XPath[3] filter - see Figure 10. This filter will be applied to the source content coming from DOORS[4].

To filter the list of objects to only return those where some CustomAttribute column has the value 'hardware' – use the following XPath expression:

```
attributes/attribute[@name='CustomAttribute' and text() = 'hardware']
```

This pattern can be adapted accordingly to filter by other attributes and values. Further details of the data structure to which the filter is applied are given in Appendix A.
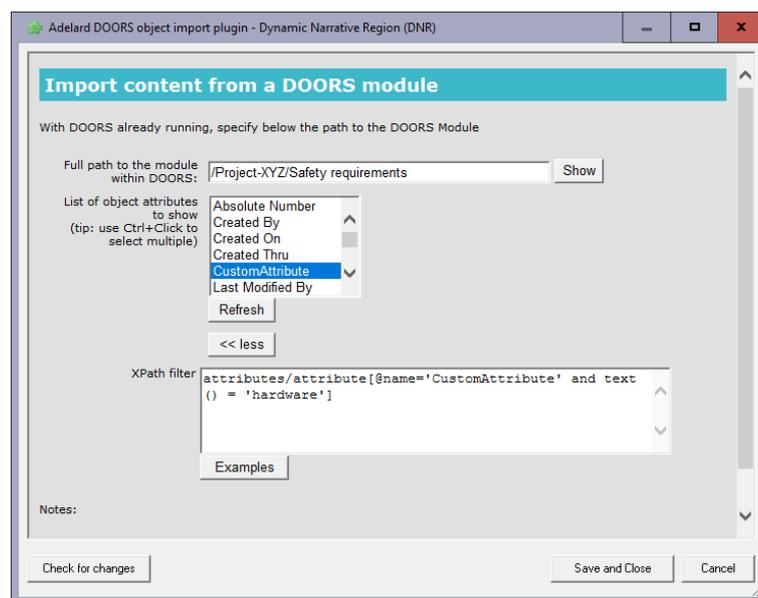


**Figure 10: Setting a filter on the results returned from DOORS**

This will then only show a subset of objects in the results – see Figure 11.

---

[3] For further information about the XPath filtering language, consult online resources.

[4] You can also filter the source information by applying a filter within the DOORS application.
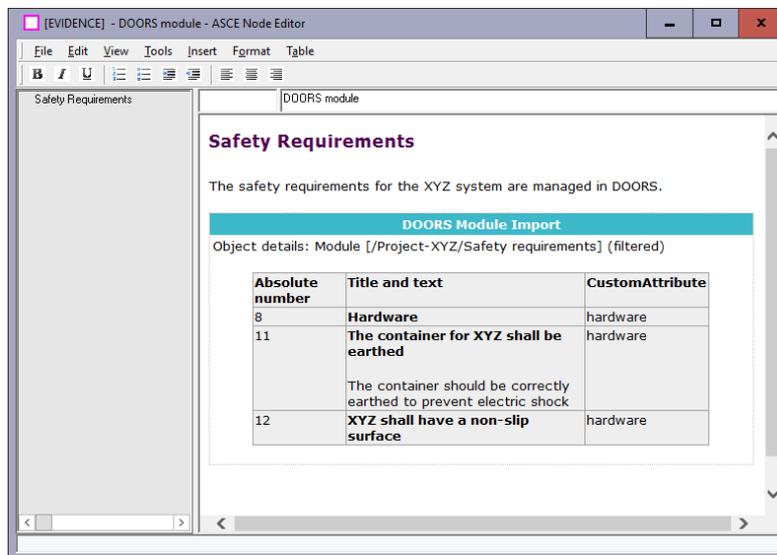
**Figure 11: Filtered results showing a subset of the information**

## 3.3    Creating an internet link to a DOORS object

DOORS Objects have a unique URL. You can also copy and paste these into the ASCE node narrative, using the standard Internet link feature.

To discover each object's URL within DOORs, view its properties. The URL is shown selected in the following screenshot from DOORS – see Figure 12.
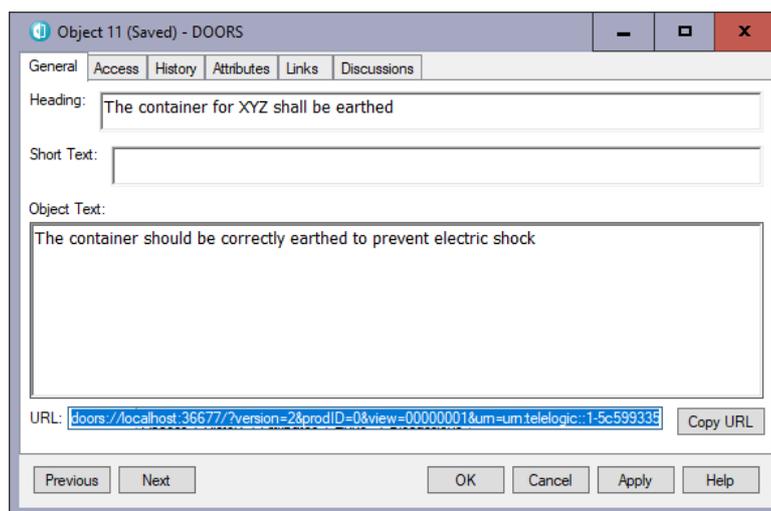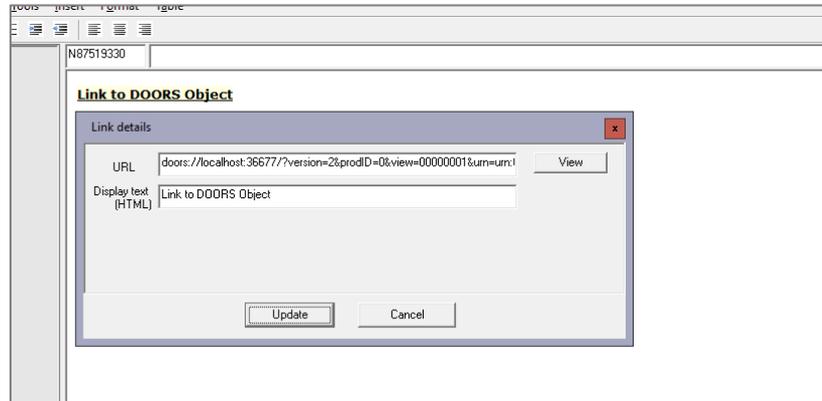


**Figure 12: Object unique URL in DOORS**

From the node editor, these URLs can be linked to as a standard hyperlink – see Linking to doors://Figure 13.

Linking to doors://Figure 13: URL in the ASCE node narrative

## Appendix A
## XML structure used for filtering module listing results

When filtering the results of a module (see Section 3.2.1) the XPath filter is applied to an XML object assembled by the plugin. If the XPath expression matches the object it will be shown in the results.

Objects returned in a module listing have the following structure, with grey data slots populated according to the content of the DOORS object:

```xml
<!--XML fragment to which XPath filter may be applied-->
<doors-object absolute-number="absoluteNumber">
   <object-heading>objectHeading</object-heading>
   <object-text>objectText</object-text>
   <object-short-text>objectShortText</object-short-text>
   <attributes>
      <!-- one entry for each attribute of the object-->
      <attribute name="attributeName">attributeValue</attribute>
   </attributes>
</doors-object>
```

As an example, an actual object in a module having two attributes "Attribute A" and "Attribute B" might look as follows:

```xml
<doors-object absolute-number="8">
   <object-heading>Heading of a requirement in DOORS</object-heading>
   <object-text>Requirement text here</object-text>
   <object-short-text>Requirements short text here</object-short-text>
   <attributes>
      <attribute name="Attribute A">Value of Attribute A</attribute>
      <attribute name="Attribute B">Value of Attribute B</attribute>
   </attributes>
</doors-object>
```