ADOIT
Enterprise Architecture Suite

ADOIT®
Version 18.1

Excel Interface Manual

BOC Group
Design your Enterprise

BOC Group Addresses:

Hotline: https://support.boc-group.com

Homepage: https://www.boc-group.com

The ADOIT® 18.1 Excel Interface Manual

Table of Contents

I Welcome to the Excel Interface Manual

ADOIT provides a configurable Excel interface for quick data acquisition. Via the Excel interface, you can import repository objects with their attributes and relations from an Excel file (XLS or XLSX format). For this process, the structure of the Excel file is described in an XML configuration file.

In this part of the manual, you will learn how to use the Excel interface. The following topics are outlined:

- **Structure of the Excel File**
  An overview of the structure of an Excel file for importing objects
- **Create Configuration**
  Learn how to create an XML configuration file which contains the mapping of objects from the Excel file to the ADOIT metamodel.
- **Perform Excel Import**
  Learn how to import objects from Excel

To successfully create a configuration, you need access to the Metamodel Management. The language independent name of object types, attributes etc. can be found there and you need those for the configuration.

1 Structure of the Excel File

An Excel file for importing repository objects is structured as follows:

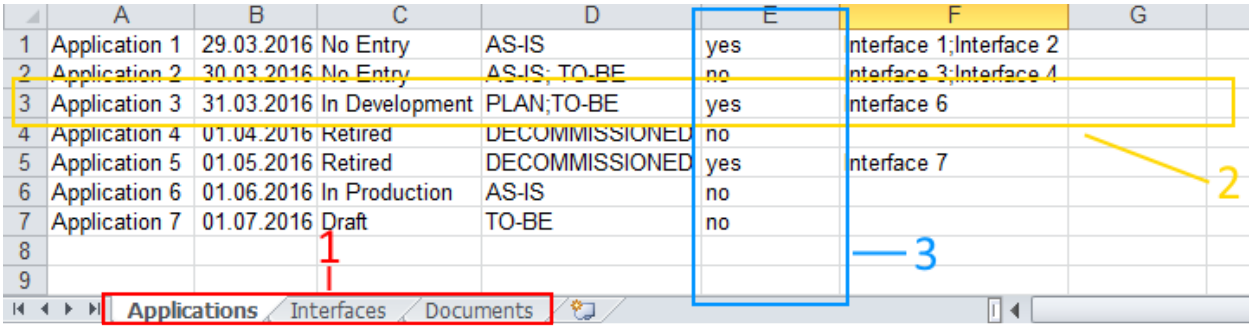


Fig. 1: Example of an Excel File with Objects and Their Attributes

- The Excel file can contain any number of sheets
- Each sheet contains only objects of one type (1)
- Each row in the sheet contains one object (2)
- Each column holds an object attribute or a relation to another object (3)
- A unique identifier (name, ID etc.) is needed for each object
- The following attribute types can be imported: `simple`, `date`, `enum`, `treeenumlist`, `enum_list`, `bool`, `relation` and `file_pointer`

If you are using the ArchiMate Application Library, one or more sample configurations will already be provided for you in the ADOIT Administration. For each sample configuration, a suitable Excel template file is included. The template is available for download in the Excel import dialog in the ADOIT.

2 Create Configuration

An XML configuration file contains the mapping of objects from the Excel file to the ADOIT metamodel.

2.1 <config>

The XML file starts and ends with a `<config>` tag:

XML Example

<config>

...

</config>

2.2 <sheet>

In the XML file, each sheet to be imported from the Excel file is described in a separate `<sheet>` tag. The following attributes must be defined for each `<sheet>`:

Attribute name	Description	Sample values
`name`	Name of the sheet in Excel	`Applications`
`class_name`	Language independent name of the object type	`[C_APPLICATION]`
`id`	Specifies the number of the column which uniquely identifies an object. [Optional] A comma separated list of column numbers which will be used to identify the object.	`1`
`data_row`	Specifies the first row containing an object	`1`
`unique_name` [Optional]	Set to `true` to prevent the creation of objects of the same type with the same name in the same language context	`[true | false]`

2.2.1 Unique Object Identification by ID

The field `id` defines which object attribute in the Excel file uniquely identifies an object.

Typical attributes for unique object identification are the attribute 'Name' or dedicated 'ID' attributes. The use of a dedicated 'ID' attribute offers the advantage that objects can be renamed in third-party systems over the course of time, but will still be identified as identical objects during import.

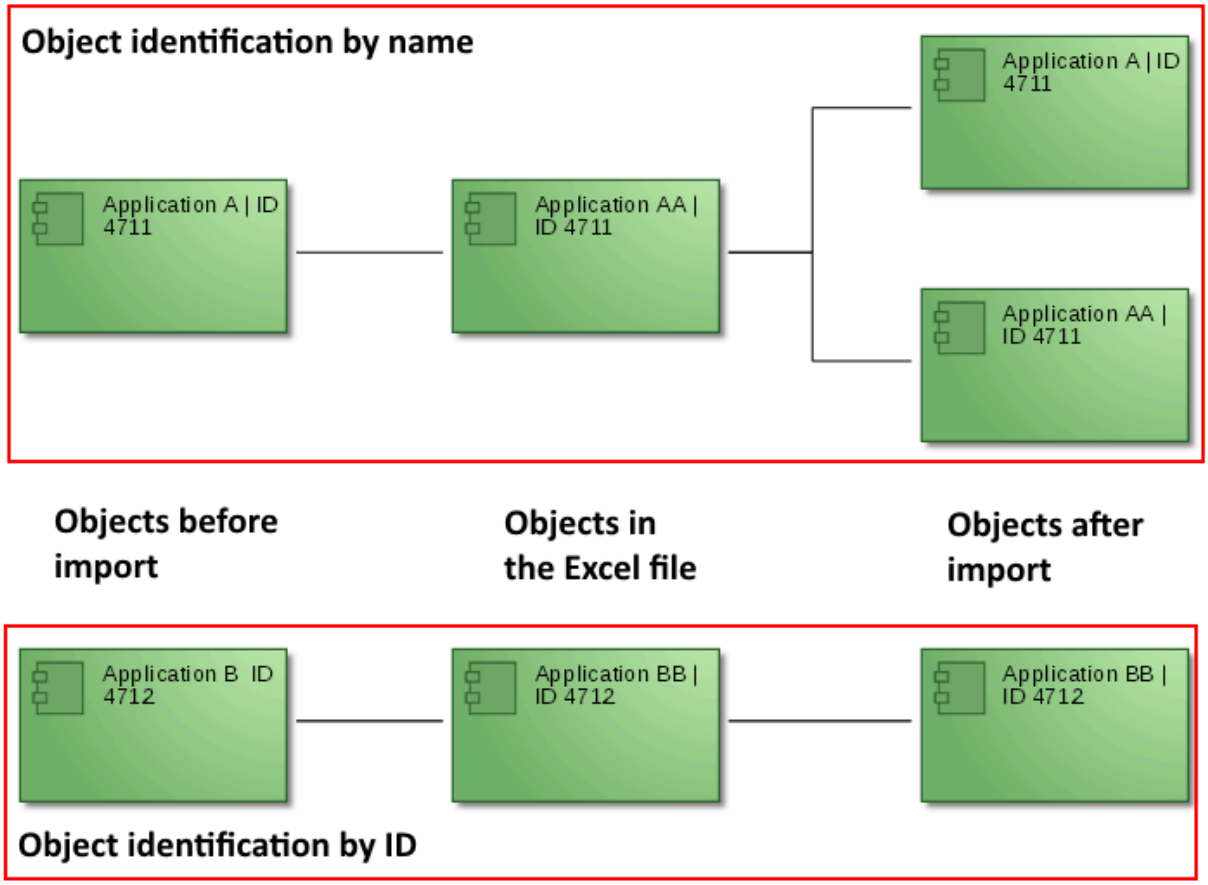
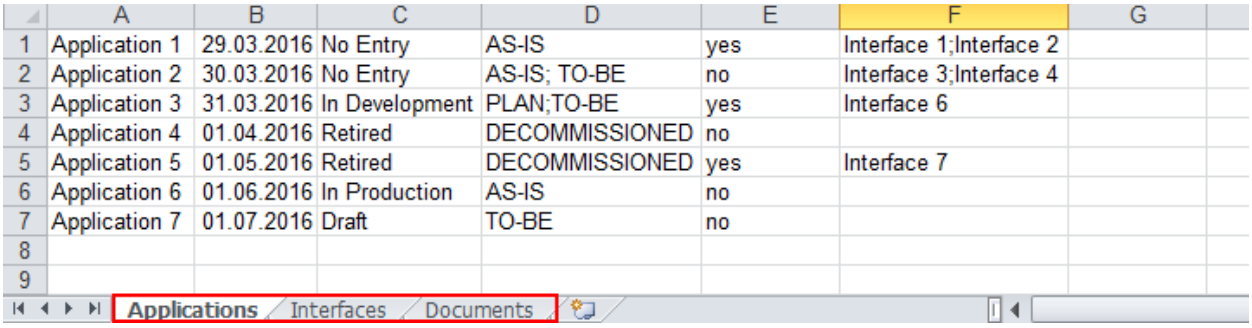


Fig. 2: Unique object identification by 'ID'

2.2.2 Example

You want to import three sheets containing objects of the type *Application*, *Interface* and *Document*:



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Fig. 3: Import of a Sheet

Below is an excerpt from the XML file with the sections for importing the sheets:

XML Example

```
<config>
<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">
...
</sheet>
<sheet name="Interfaces" class_name="C_INTERFACE" id="1" data_row="1">
...
</sheet>
<sheet name="Documents" class_name="C_DOCUMENT" id="1" data_row="1">
...
</sheet>
</config>
```

2.3 <attribute>

In the XML file, each object attribute to be imported from the Excel file is described in a separate `<attribute>` tag. The following attribute types can be imported:

Attribute type	Description
`simple`	Stores textual values or numerical values
`date`	Stores dates
`enum`	Assignment of a value to an element of a list
`enum_list`	Assignment of values to the elements of a list

`treeenumlist`	Assignment of values to the nodes of a tree
`bool`	Stores Boolean values
`relation`	Stores relationships with other objects
`file_pointer`	Stores links to external files

2.3.1 <attribute type="simple">

Textual values and numeric values can be imported using the `simple` attribute type. The imported values can be assigned to the following attribute types in the ADOIT metamodel:

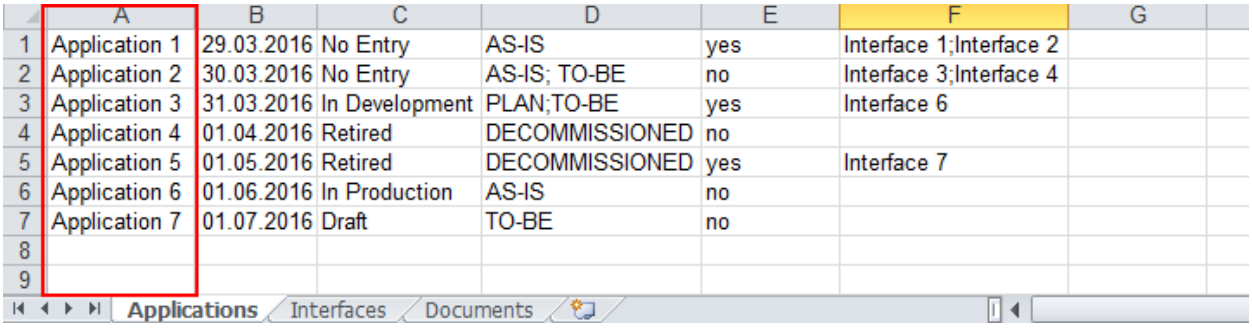
- ADONIS String (ADOSTRING)
- Integer (INTEGER)
- Double (DOUBLE)
- Short String (SHORTSTRING)
- Long String (LONGSTRING)
- Unsigned Integer (UNSIGNED INTEGER)
- String (STRING)

To do this, the following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `simple`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English. Possible values depend on the available content languages.
`column`	Specification of the column number of the object attribute in the Excel file

2.3.1.1 Example

You want to import the attribute *Name*, which is contained in the properties of the object type *Application*.



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Fig. 4: Import of a `simple` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

<config>

<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">

<attribute name="NAME" type="simple" context="en" column="1"/>

...

</sheet>

...

</config>

2.3.2 <attribute type="date">

Date values can be imported using the `date` attribute type. The imported values can be assigned to the following attribute types in the ADOIT metamodel:

- Date (DATE)
- Coordinated Universal Time (UTC)

Date values are all imported as 00:00 UTC. To ensure any customised scheduled jobs run based on local time, you can configure an offset to UTC.

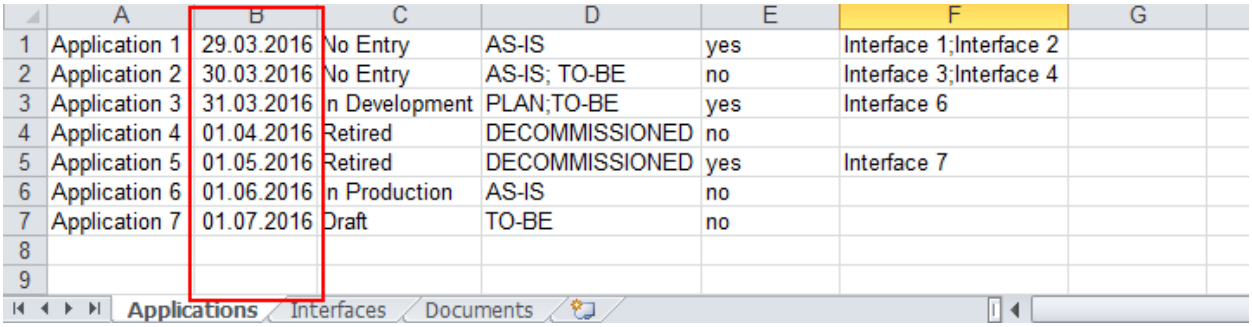
The following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `date`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English.

	Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the object attribute in the Excel file. This column should be formatted as a date in the Excel file.
`utc_offset_hours` [Optional]	Specification of the offset in hours from Coordinated Universal Time (UTC) to local time, e.g. "1" for Central European Time (CET) or "-5" for Eastern Standard Time (EST).
`utc_offset_minutes` [Optional]	Specification of the offset in minutes from Coordinated Universal Time (UTC) to local time, e.g. "30" for Indian Standard Time (IST) for a total time offset of +05:30 (if `utc_offset_hours` is set to "5").

2.3.2.1 Example

You want to import the attribute *Production Date*, which is contained in the properties of the object type *Application*.



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Applications | Interfaces | Documents

Fig. 5: Import of a `date` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

```
<config>
<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">
...
<attribute name="A_VALID_FROM" type="date" context="en" column="2"/>
...
</sheet>
...
```

```
</config>
```

2.3.3 <attribute type="enum">

Attribute values can be imported and assigned to the elements of a list using the `enum` attribute type. The imported values can be assigned to the following attribute type in the ADOIT metamodel:

- Enumeration (ENUM)

An ENUM attribute is a list of predefined values (e.g 'Low', 'Medium' and 'High'). Only one value can be selected from the list.

Via the Excel interface, you can assign values from an Excel file to the values in the list. You can freely define which value from the Excel file is transferred to which value in the list. To do this, the following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `enum`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English. Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the object attribute in the Excel file
`domain_mapping`	Definition of the mapping between the values of the Excel file and the elements of the list (see below the table for details).
`separator_domain_mapping` [Optional]	Define the separator between multiple values within the `domain_mapping` attribute. For example, if the values are listed as "No Entry@v0,..." in the Excel file, the value of the separator is "@".

2.3.3.1 domain_mapping

Each element in a list in ADOIT has a language independent name - e.g. v0, v1, v2, etc. The values in the Excel file must be mapped to the language independent names of the elements of the list. This is done using the `domain_mapping` attribute. A corresponding value from

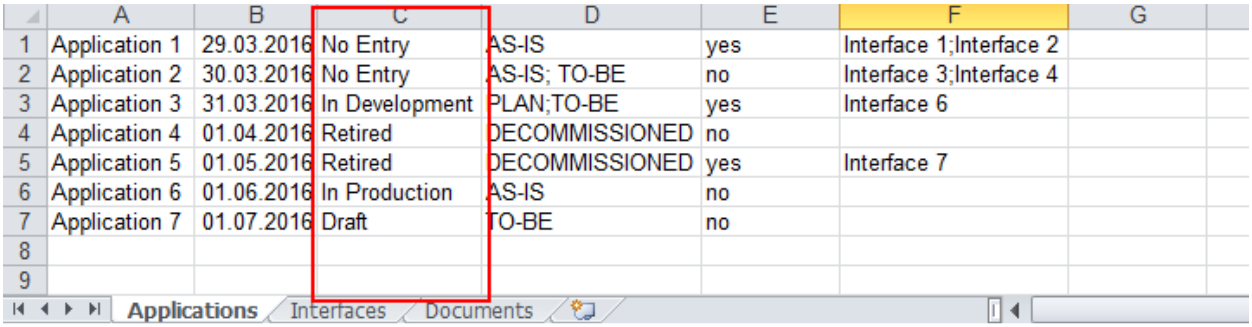
the list is assigned to each value from the Excel file: This attribute has to contain a comma-separated list of value pairs:

```
Domain_mapping="<Excel Value 0>,v0,<Exce
l Value 1>,v1,<Excel Value 2>,v2,…"
```

The values from the Excel file and the *language dependent values* of the ENUM attribute do not have to be the same. For example, you can assign the Excel value 'Pink' to the predefined value 'Red' in the list.

2.3.3.2 Example

You want to import the attribute *Lifecycle State*, which is contained in the properties of the object type *Application*.



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Applications | Interfaces | Documents

Fig. 6: Import of an `enum` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

<config>

<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">

...

<attribute name="A_LIFECYCLE_STATE" type="enum" context="en" column="3" domain_mapping="Kein Eintrag,v0,Entwurf,v1,In Entwicklung,v2,In Produktion,v3,Stillgelegt,v4"/>

...

</sheet>

...

</config>

2.3.4 <attribute type="enum_list">

Attribute values can be imported and assigned to the elements of a list using the `enum_list` attribute type. The imported values can be assigned to the following attribute type in the ADOIT metamodel:

- Enumeration List (ENUMLIST)

An attribute of the type `enum_list` is a list of predefined values (e.g 'Low', 'Medium' and 'High'). In contrast to attributes of the type ENUM, multiple values from the list can be selected at the same time.

You can define freely which value from the Excel file is transferred to which value in the list. To do this, the following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `enum_list`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English. Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the object attribute in the Excel file
`separator`	Define the separator between multiple values in the Excel file which should be assigned to the list. In this way, you can assign multiple values to the ENUMLIST attribute at the same time. For example, if the values are listed as "Variant 1;Variant 2" in the Excel file, the value of the separator is ";".
`domain_mapping`	Definition of the mapping between the values of the Excel file and the elements of the list (see <attribute type='enum'> for details).
`separator_domain_mapping` [Optional]	Define the separator between multiple values within the `domain_mapping` attribute. For example, if the values are listed as "No Entry@v0,..." in the Excel file, the value of the separator is "@".

2.3.5 <attribute type="treeenumlist">

Attribute values can be imported and assigned to the nodes of a tree using the `treeenumlist` attribute type. The imported values can be assigned to the following attribute type in the ADOIT metamodel:

- ENUMLIST_TREE

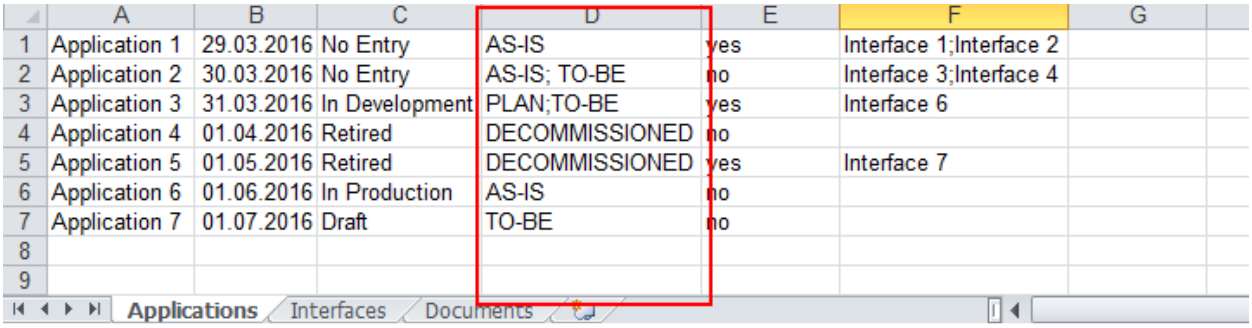
Attributes of the type ENUMLIST_TREE are trees with predefined values (e.g. "Variant 1", "Variant 2" and "Variant 3"). Multiple values from the tree can be selected at the same time. The categories filter in ADOIT allows filtering objects according to these values (= categories).

You can define freely which value from the Excel file is transferred to which node. To do this, the following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `treeenumlist`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English. Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the object attribute in the Excel file
`separator`	Define the separator between multiple language independent names of tree nodes in the Excel file. In this way, you can assign multiple values to the 'ENUMLIST_TREE' attribute at the same time. For example, if the values are listed as "PLAN;TO-BE" in the Excel file, the value of the separator is ";".

2.3.5.1 Example

You want to import the attribute *Variants*, which is contained in the properties of the object type *Application*.



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Applications | Interfaces | Documents

Fig. 7: Import of a `treeenumlist` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

<config>

<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">

...

<attribute name="A_ARCHITECTURE_TREE" type="treeenumlist" context="en" column="4" separator=";"/>

...

</sheet>

...

</config>

2.3.6 <attribute type="bool">

Boolean values ('TRUE' and 'FALSE') can be imported using the `bool` attribute type. The imported values can be assigned to the following attribute type in the ADOIT metamodel:

- Bool (BOOL)

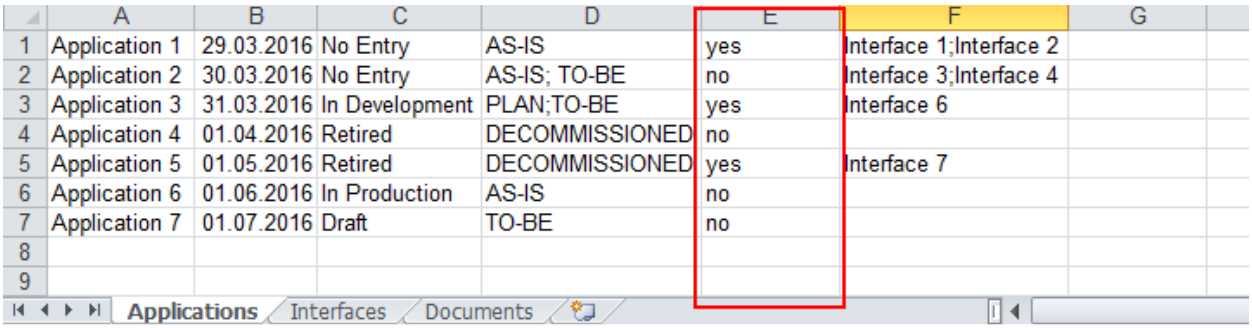
To do this, the following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `bool`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English.

	Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the object attribute in the Excel file
`bool_mapping`	Definition of the mapping between the values of the Excel file and the values of the object attribute 'TRUE' and 'FALSE'. This attribute contains two values, separated by a comma. The first value is mapped to the boolean value 'TRUE', the second to 'FALSE'.

2.3.6.1 Example

You want to import the attribute *Action Required*, which is contained in the properties of the object type *Application*.



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Applications | Interfaces | Documents

Fig. 8: Import of a `bool` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

<config>

<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">

...

<attribute name="A_NEED_FOR_ACTION" type="bool" context="en" column="5" bool_mapping="ja,nein"/>

...

</sheet>

...

</config>

2.3.7 <attribute type="relation">

Relations between objects can be established via the attribute type `relation`. In the ADOIT metamodel, these relations are represented by relation classes.

The following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the relation class to which the referenced objects will be assigned
`type`	Specification of the attribute type - here: `relation`
`context`	Specification of the language of the attribute which uniquely identifies the referenced object, e.g. "de" for German and "en" for English. Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the relation in the Excel file
`separator`	Define the separator between multiple referenced objects in the Excel file. If the referenced objects are listed as "interface1;interface2;interface3", the value of the attribute is ";".
`attr_separator` [Optional]	Define the separator between multiple attributes values used to identify the referenced objects. If the referenced objects are listed as "interface1@description1;interface2@description2", the value of the attribute is "@".
`lookup_attr_name`	Language independent name of the attribute which uniquely identifies the referenced object. This does not have to be set to the "NAME" attribute. [Optional] A list of language independent attribute names which will be used to identify the referenced object.
`direction`	Determine whether the relation should be interpreted as an incoming reference or outgoing reference. Possible values are "from" (incoming reference) and "to" (outgoing reference).
`target_class`	Language independent name of the referenced object type

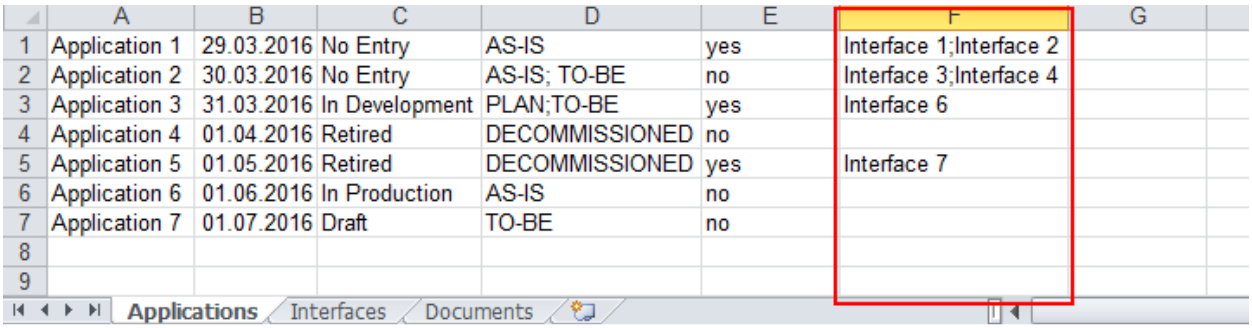
`strategy`	The attribute value determines how to handle the conflicts regarding the relations during the Excel import. See section Import Conflict Management for more information. Set to `overwrite` by default.

2.3.7.1 Incoming References vs. Outgoing References

For each reference a decision on the source and the target object has to be made. Taking the perspective of the source object, the relation can be classified as an outgoing reference (symbol). Taking the perspective of the target object, the relation can be classified as an incoming reference (symbol).

2.3.7.2 Example 1

You want to create relations in the attribute *Provided Interfaces*, which is contained in the properties of the object type *Application*.



	A	B	C	D	E	F	G
1	Application 1	29.03.2016	No Entry	AS-IS	yes	Interface 1;Interface 2	
2	Application 2	30.03.2016	No Entry	AS-IS; TO-BE	no	Interface 3;Interface 4	
3	Application 3	31.03.2016	In Development	PLAN;TO-BE	yes	Interface 6	
4	Application 4	01.04.2016	Retired	DECOMMISSIONED	no		
5	Application 5	01.05.2016	Retired	DECOMMISSIONED	yes	Interface 7	
6	Application 6	01.06.2016	In Production	AS-IS	no		
7	Application 7	01.07.2016	Draft	TO-BE	no		
8							
9							

Applications | Interfaces | Documents

Fig. 9: Import of a `relation` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

<config>

<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">

...

<attribute name="RC_PROVIDED_INTERFACES" type="relation" context="en" column="6" separator=";" lookup_attr_name="NAME" direction="TO" target_class="C_INTERFACE"/>

...

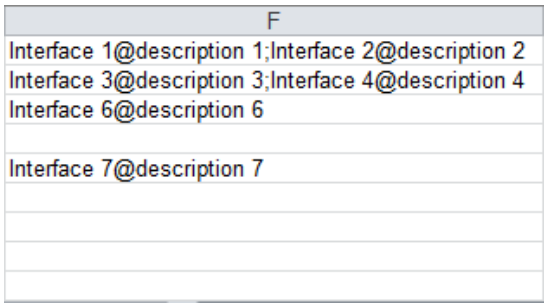
</sheet>

...

```
</config>
```

2.3.7.3 Example 2

As in Example 1 above, however the content of the cell in the Excel file in column F looks like this:



F
Interface 1@description 1;Interface 2@description 2
Interface 3@description 3;Interface 4@description 4
Interface 6@description 6
Interface 7@description 7

Fig. 10: Import of a `relation` Attribute

That is, the referenced objects are uniquely identified by two attributes.

Below is an excerpt from the XML file with the relevant section

XML Example

```
<config>
<sheet name="Applications" class_name="C_APPLICATION" id="1" data_row="1">
...
<attribute name="RC_PROVIDED_INTERFACES" type="relation" context="en" column="6" separator=";" attr_separator="@"; lookup_attr_name="NAME,DESCRIPTION" direction="TO" target_class="C_INTERFACE"/>
...
</sheet>
...
</config>
```

2.3.8 <attribute type="file_pointer">

Relations to external objects or programs can be established via the attribute type `file_pointer`. The imported values can be assigned to the following attribute type in the ADOIT metamodel:

- File Pointer (FILE_POINTER)

To do this, the following attributes must be defined for the tag `<attribute>`:

Attribute type	Description
`name`	Language independent name of the object attribute to which the imported value will be assigned
`type`	Specification of the attribute type - here: `file_pointer`
`context`	Specification of the language of the attribute value, e.g. "de" for German and "en" for English. Available languages depend on the Application Library and licence.
`column`	Specification of the column number of the object attribute in the Excel file
`separator`	Defines a separator between program name and file reference in the Excel file. If the separator is empty, then the cell content will be imported as a file reference and program name will be kept empty.

2.3.8.1 Accessibility of Links to Files

In modern browsers, links to files on a local hard drive, UNC links to files on a server and also links to network drives are not accessible for security reasons. Links to files on other websites are accessible normally.

2.3.8.2 Example

You want to create relations to external documents in the attribute *Referenced Document*, which is contained in the properties of the object type *Document*:

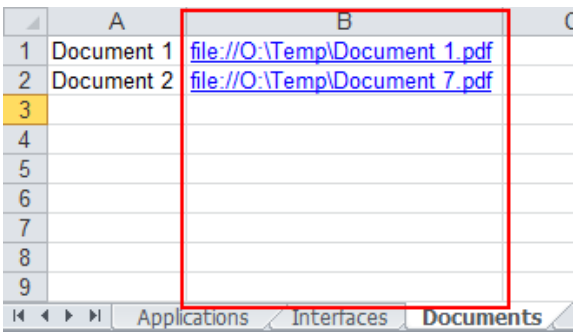


Fig. 11: Import of a `file_pointer` Attribute

Below is an excerpt from the XML file with the relevant section

XML Example

```
<config>
<sheet name="Documents" class_name="C_DOCUMENT" id="1" data_row="1">
...
<attribute name="A_REFERENCED_DOCUMENT" type="file_pointer" context="en" column="2"/>
...
</sheet>
...
</config>
```

3 Perform Excel Import

Performing an Excel import consists of the following steps:

3.1 Import Configuration

In order to use a specific configuration, you have to import the XML configuration in the ADOIT Administration first:

1. Go to the *Settings* page.
2. Right-click *Excel import*, and then click *Create*.
3. In the *Configuration name* box, enter a name for your configuration, and then click *OK*.
4. Click *Import* and upload the XML configuration file. The content of the configuration file will be displayed in the *Setup* box.
5. Click *Save*.

The configuration is saved. You can now import objects from any Excel file that corresponds to the configuration.

3.2 Import Excel File Template

For each configuration you can import a suitable Excel file as a template in the ADOIT Administration.

1. Go to *Settings* > *Excel Import*.
2. Select the configuration for which you want to import a template.
3. Click *Import Excel Template* and upload the Excel file. The template's name will be displayed in the Template box.
4. Click *OK*.

The configuration is saved. The template is now available for download in the Excel import dialogue in ADOIT. Users can download the template, capture objects in it, and then import the objects.

3.3 Import Objects from Excel in ADOIT

Once you have imported an XML configuration file, you can import objects from any Excel file that corresponds to the configuration in ADOIT:

1. Select the object group in which the new objects shall be created in the Object Catalogue.
2. Right-click the object group, point to *Import/Export*, and then click *Import objects from Excel*. A dialogue window opens.
3. Select the desired configuration from the drop-down list *Select configuration*.

4. Enter the path and name of the import file into the field *Excel file* (either manually or via the *Select file…* support dialogue ➕).
5. Click *Next*. Another dialogue window containing an *Import preview* opens. Review the information to verify that the collected data is processed properly.
6. Confirm with *Import*. The data is imported and a confirmation box appears. Click *Details* to verify if the import was successful. Then close the box.

Additionally, you can:

- Download a suitable Excel file as a template. In order to do so, click the *Download template* button 📥. You can capture objects with this template, and then import the objects.

3.4 After the Import

After a successful import, you can find the imported objects in the Object Catalogue:

- Objects which already existed in the repository before the import and were only updated can be found in their usual location in the Object Catalogue.
- When you import objects for the first time, the imported objects will be saved in the object group "_*Unassigned*\<*user name*>*Excel-Import* <*date*>\<*object type*>".

<*user name*> is the user who was used for the import, e.g. "Admin".

II Appendix

In the Appendix, the following topics are outlined:

- **Import Conflict Management**
  Conflict management strategies which are applied when importing objects
- **Display Language Independent Names of Metamodel Elements**
  Display language independent names of object types, relation classes and attributes

4 Import Conflict Management

The following conflict management strategies are applied when importing objects:

- Attributes that are not defined in the Excel file respectively not defined in the XML file are not changed by the Excel import.
- `Simple` attribute values will be overwritten with the values from the Excel file (even if the Excel file contains empty values).
- The values of `date` attributes will be overwritten if the Excel file contains valid data. The values will be removed if the date cell in the Excel file is empty.
- If the parameter `strategy` is set to `overwrite`, relations will be overwritten with the values from the Excel file:
  - New relations will be created.
  - Relations which are not present in the Excel file will be removed.
  - If the relation cell is empty, then all the relations of the configured type will be removed from the object.

  If the parameter `strategy` is set to `add`, existing relations will not be removed. New relations will be created.
- Handling of referenced objects when creating relations:
  - If the referenced object exists in the repository, the relation will be created.
  - If the referenced object exists in the Excel file, the referenced object and the relation will be created
  - If the referenced object neither exists in the Excel file nor in the repository, the relation will not be created.
  - If the referenced object exists in the Excel file, but could not be created, the relation will not be created.
  - If it is unclear which object should be referenced because there are multiple potential targets, the relation will not be created.

5 Display Language Independent Names of Metamodel Elements

To successfully create a configuration, you need access to the *Properties* page in the ADOIT Administration. The language independent name of object types, attributes etc. can be found there and you need those for the configuration.

To display the language independent names of metamodel elements:

1. Open the *Properties* page.
2. Select the desired object type from the *Select class or model type* list at the top of the page.
   - The language-independent name of the object type appears to the right of the list.
   - The properties of the object type are displayed in the workspace (organised into chapters). The language-independent names of attributes and relations appear in the *Language-independent* column.

Index

T

www.boc-group.com

BOC Group

Athens • Berlin • Dublin • Madrid • Paris • Vienna • Warsaw • Winterthur