SPAIX® 5

PRODUCT DOCUMENTATION
Spaix 5 ProjectAdapter XLS (WIN)

Last updated: 11|2022

info@vsx.net | www.vsx.net

CONTENTS

1 INTRODUCTION

The Spaix Excel ProjectAdapter provides tabular presented information from an Excel file to be used in Spaix as project as well as to save the results of pump selections and configurations into excel files. The **import** from Excel to Spaix offers the possibility to transfer basic project and **selection specifications**. The **export** allows to export any data which is available via Spaix keywords from Spaix into Excel files. The following documentation refers to the windows version of Spaix.

2 GENERAL PROCEDURE

The Excel project adapter import and export function are available via the Spaix selection program. When properly configured the Excel adapter import/export options can be found in the burger menu under **Open** and **Export**. For the definition which values in Spaix correspond to which cell in excel at least two files are required: an XML-scheme and an Excel-source respectively a Excel-output-template file. The XML is adapted to the intended structure of the Excel file and contains references to the information which should be passed to or output from Spaix.

The **import** is based on XML parser file (**XlsAdapter_Parser.xml**). The definition inside must meet the corresponding Excel which holds the information to be imported. Please note that this file must be available in Spaix document directory exactly with this name.

For the **export** at least one **XML** template file and a corresponding **Excel** output template file must be specified. In contrast to the import for the export also several different schemes for creating different variants of Excel files can be defined. XML and Excel files for the export can also be located outside the Spaix DocumentRoot directory under an arbitrary file name. File names and if different from DocumentRoot also the path must always be specified in an additional file (**vsAdapter.xml**). If multiple styles are defined for export users are offered the list of available styles from which one can be selected.

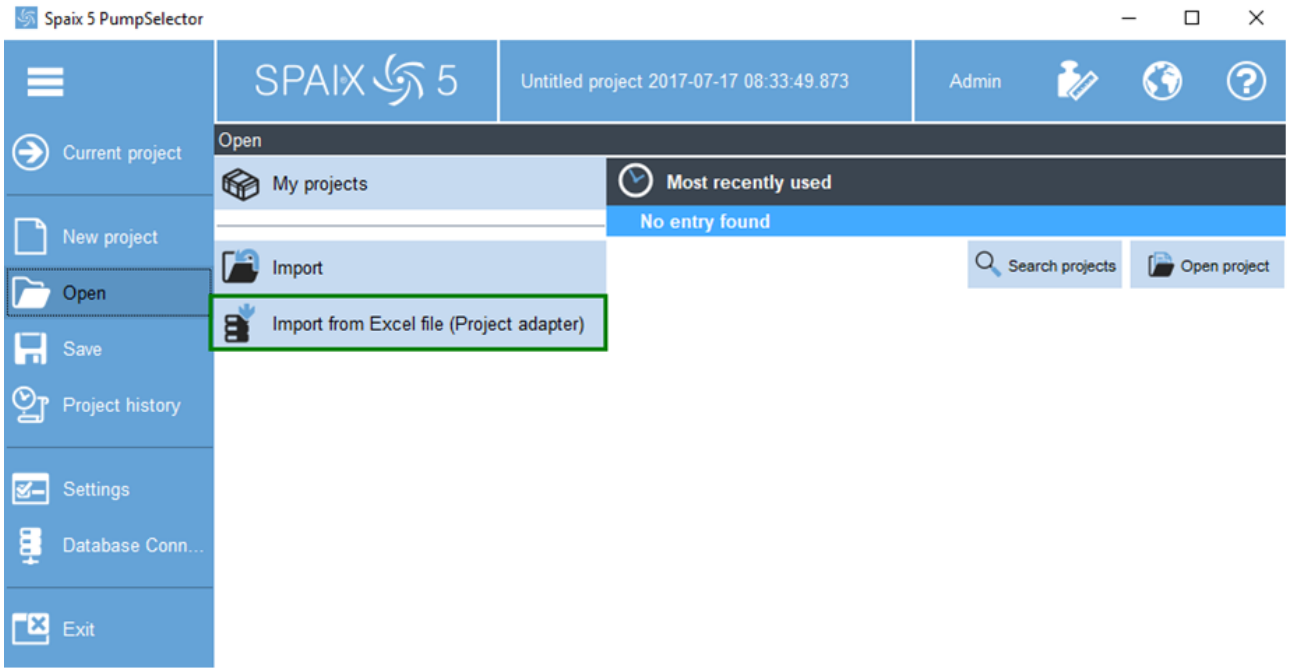

Figure 1: The menu **Import / Export** *in the Spaix selection program*

3 STRUCTURE OF AN EXCEL FILE

The sole purpose of creating an Excel template is to present a practical and attractive layout to users who enter or read data in Excel. This usually includes layout aspects for corporate design like logo and colors but also frames and column widths for defining table areas like for project, contact and pump data.

Excel files can be designed almost without restriction according to one's own requirements.

The utilized sheet of the Excel template has to be always the first excel sheet. Also this first sheet must not be a “hidden sheet” because this is not supported by the export.

4 STRUCTURE OF THE PARSER XML FILE

For building an XML structure, basic knowledge of the file format is recommended (tree structure, elements, attributes). To get started with the topic, there are numerous tutorials on the web, for example: https://www.w3schools.com/xml/

With XML syntax, note that some characters must be rewritten/encoded if they are to be used in the normal text between the tags or in tag-names itself:

Zeichen	Notation in XML
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

4.1 GENERAL STRUCTURE

The XMl file is a nested structure of different elements. It should be coded Unicode. The root-tag is defined by an object element (<DataItem>).

Right beyond the root-tag can be an optional paragraph containing information about formatted areas (see following chapter Section Format), followed the elements for accessing the data. The single elements and attributes which are used in the parser scheme are described more detailed in the following section.

4.2 SECTION FORMAT

If the Excel template file contains formatted areas which are supposed to be templates for the presentation of the data, they have to be stated here. The paragraph is marked through the XML Element <FormatTemplates> in the template file. The area information of the existing model block have to be stated here.

The following syntax is to be followed:

Element	Usage
XLSTemplate	Description of a model block

Table 1: Used XML Elements in the section scheme

The following XML attributes are used to describe the scheme:

Format-Element	Attribut	Verwendung
XLSTemplate	Name	Clearly defined name for the model block
XLSTemplate	RelRow	First row of the model block
XLSTemplate	OffsetRow	Number of rows of the model block

Table 2: Used XML Attributes in the scheme

Example:

```
<DataItem>
  <FormatTemplates>
    <XLSTemplate Name="MainIten" RelRow="12" OffsetRow="3">
    <XLSTemplate Name="SubItem" RelRow="15" OffsetRow="1">
  </FormatTemplates>
```

Figure 2: Example for a formatting paragraph

The shown example could lead to the following XML cut:

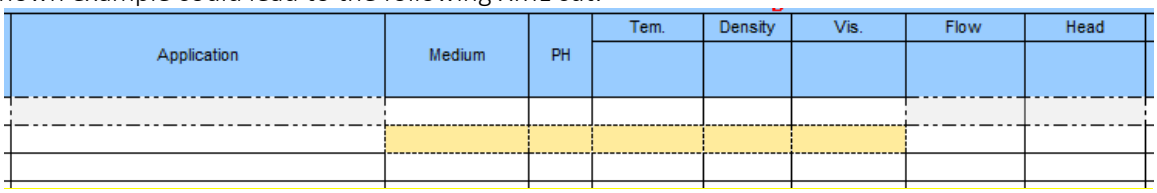

Figure 3: Example for a formatted Block on the Excel template

The RelRow + Offset of an XLSTemplate definition does not correspond to the first row in the resulting Excel.

In the resulting Excel, the row(s) which are defined as template block are output as hidden row(s) (from version 5.2022.3). This is necessary because otherwise it led to corrupted Excel files if the row(s) were deleted and at the same time other Excel logics (own Excel code or queries) are used which refer to these rows.

4.3 SELECTION SCHEME

4.3.1 OVERVIEW OF THE USED XML ELEMENTS

The following XML elements are used to describe the scheme

Element	Use
DataList	Description for a list of objects
DataItem	Description for a single object
DataValue	Description for a property

Table 3: Used XML elements in the scheme

4.3.2 OVERVIEW OF THE USED XML ATTRIBUTES

The following XML attributes are used to describe the scheme:

Scheme-Element	Attribute	Use
DataList	ListRef	Object reference for a list object
DataList	ListItemRef	Object reference for a list element
DataList	RelCol	Column of first appearance of the list object relating to the superior anchor
DataList	FormatTemplate	Reference to the format block to be used during export (see **Fehler! Verweisquelle konnte nicht gefunden werden.**)
DataItem	Required	If *true*, mandatory information exist for the element
DataValue	ValueRef	Formula to a Spaix property
DataValue	RelRow	Relative line space to the superior anchor
DataValue	RelCol	relative column space to the superior anchor
DataValue	Required	If *true*, the object property is a mandatory field
DataValue	Exist	If true, the object property will be set/checked on a certain value (defined in ValueTag) and displayed as assignment value.
DataValue	ValueTag	Value of an object property to be set/checked

Tabele 4: Used XML Attributes in the scheme

Example:

```
<DataItem>
 <DataList ListRef ="{PKI}" ListItemRef="[AW]" RelRow="12" RelCol="1" OffsetRow="1">
   <DataItem Required="1">
     <DataValue ValueRef="{AW{$Index=%}.Name}" RelRow="0" RelCol="0" Required="1"/>
     <DataValue ValueRef="{AW{$Index=%}.FREQ}" RelRow="0" RelCol="6"/>
     <DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT.DP_H1}" RelRow="0" RelCol="2"/>
     <DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT.DP_Q1}" RelRow="0" RelCol="3"/>
```

Figure 4: Example for a section of a scheme

4.3.3 SPAIX REFERENCES

The Spaix references contain the Spaix formulas in curly brackets. The formulas which are defined under the XML attribute ValueRef lead to a specific property in Spaix which leads with optimal formatting instructions to the output of a value. For a detailed description of Spaix formulas please refer the document Spaix 5 Formulas.pdf.

The template can define the output of single objects which can for example contain general information about the project or offer, sender and receiver. They can be used for the transfer of information to Spaix but also in the opposite direction.

Example:

```
<DataItem>
        <DataValue ValueRef="{PKBY.PKADD.FIRM}" RelRow="3" RelCol="3"/>
        <DataValue ValueRef="{PKTR.PCKBUSIPID}" RelRow="5" RelCol="3"/>
        <DataValue ValueRef="{PKTR.DATELAST}" RelRow="6" RelCol="3"/>
      </DataItem>
```

Figure 5: Example of a scheme for in- and output of project data

PKBY	… customer	PKTR	… Business transaction
PKADD	… Address	PCKBUSIPID	… Business process ID
FIRM	… Company	DATELAST	… Last update

4.3.4 SPECIAL CHARACTERISTIC OF LISTS

In order to describe lists the XML attributes ListRef and ListItemRef are available to control the output of a list of objects. Lists are always necessary if several elements are addressed of which it is not clear if, in which number or from which exact type they will be created during the selection and configuration. For example, this could be all products of an offer.

In the example of Figure 4 with ListRef={PKI}“ all elements of a request will be output.

After processing through Spaix the shown example could lead to the following Excel section:

Application	Medium	Head	Con.	NPSHa	NPSHr	Frequency
		m	%	m	m	Hz
ACP200-400.5	Kondensat	36,0	100,0		6,6	50,00
AD80-265.3	Wasser	20,20	92,10		1,25	60,00
ProjectLast	Water, pure	8,00	100,00			50,00

Figure 6: Example for a Spaix output to the scheme of figure 4

4.4 EXAMPLES MINIMAL DEFINITION

If only **project fields** are to be exported, only the DataItem element must be defined in XML.
Only valid starting from program version 5.2022.3!

Example: Excel Adapter XML Schema Minimum Definition for Project Fields

```
<?xml version="1.0" encoding="utf-8"?>
<DataItem>
  <DataItem>
    <DataValue ValueRef="{PKBY.PKADD.FIRM}" RelRow="3" RelCol="3"/>        <!-- Customer name -->
    <DataValue ValueRef="{PKBY.PKPRS.CONTACT}" RelRow="4" RelCol="3"/>     <!-- Contact name -->
  </DataItem>
</DataItem>
```

When **line items** need to be exported, the XML file needs to contain FormatTemplates definition as well as DataList definition.
DataList definition need to refer to a valid FormatTemplate and need to include RelCol.

Example: Excel Adapter XML Schema Minimum Definition for Line Items

```
<?xml version="1.0" encoding="utf-8"?>
<DataItem>
  <FormatTemplates>
    <XLSTemplate Name="LineItems" RelRow="1" OffsetRow="1"/>
  </FormatTemplates>
  <DataList ListRef ="{PKI}" ListItemRef="[AW]" FormatTemplate="LineItems" RelCol="1">
    <DataItem Required="1">
      <DataValue ValueRef="{AW{$Index=%}.PosNr}" RelRow="0" RelCol="1" RootedValue="True"/>
      <DataValue ValueRef="{AW{$Index=%}.ANZA}" RelRow="0" RelCol="2" RootedValue="True"/>
      <DataValue ValueRef="{AW{$Index=%}.CUSTPOSTXT}" RelRow="0" RelCol="3" RootedValue="True"/>
      <DataValue ValueRef="{AW{$Index=%}.PRODDESC}" RelRow="0" RelCol="4" Required="1" RootedValue="True"/>
      <DataValue ValueRef="{AW{$Index=%}.P.RESULTID}" RelRow="0" RelCol="5" ReadOnly="1" RootedValue="True"/>
      <DataValue ValueRef="{AW{$Index=%}.MED}" RelRow="0" RelCol="6" RootedValue="True"/>
      <DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT{$Index = 0}.DP_Q1}" RelRow="0" RelCol="7" RootedValue="True"/> <!-- Flow (Q) of medium-->
      <DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT{$Index = 0}.DP_H1}" RelRow="0" RelCol="8" RootedValue="True"/> <!-- Head (H) of medium-->
    </DataItem>
  </DataList>
</DataItem>
```

5 IMPORT FROM EXCEL INTO SPAIX

As explained in the chapters Introduction and General Procedure, the option for data exchange from Excel to Spaix provides the possibility to import basic data from Excel.

Basic data in the narrower sense are:

- Project settings (e.g. project name, project number)
- Contact information (address data of purchaser, supplier)
- Project items (e.g. designations, number)
- Default data (e.g. operating points, media data)

- Importing project settings and contact information creates an empty project with the imported defaults.

- Importing project settings and contact information creates an empty project with the imported defaults.

- With the import of default data, which require corresponding project positions, these data are taken over for a processing of the positions via the hydraulic selection.

When importing project positions, the intended procedure is to fill empty project positions by editing via Hydraulic selection. If default data are transferred for this purpose, they are adopted in the hydraulic selection.

Note: To specify/import fluid data, the object path must refer to the operating point data (e.g. AW.CFOPPTS.CFOPPT.MED.MEDT) so that these are transferred correctly. The abbreviated notation (AW.MEDT) can only be used for export.

Example: Excel adapter XML schema excerpt for import/export of media data

```
<DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT{$Index = 0}.MED.MED}" RelRow="0" RelCol="9" />
<DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT{$Index = 0}.MED.MEDT}" RelRow="0" RelCol="10" />
<DataValue ValueRef="{AW{$Index=%}.CFOPPTS.CFOPPT{$Index = 0}.MED.MEDD;;kg/m³;0;0}" RelRow="0"
RelCol="11" />
```

COPYRIGHT

Publisher:

VSX – VOGEL SOFTWARE GmbH
Hofmühlenstraße 4
01187 Dresden | Germany
info@vsx.net | www.vsx.net

Warranty limitations:

Although this manual was drawn up with utmost care, we are not liable for the correctness of the contents. The same applies to all names and data used in the presented examples. The information in this manual is subject to alterations without prior notice.

We are grateful for any suggestions regarding improvements as well as for any points that may be brought to our attention as well as constructive criticism.

Trademarks:

This documentation contains references to protected trademarks that are not explicitly identified within the text. Despite the fact, that protected trademarks are not explicitly identified, the rights of VSX – Vogel Software GmbH or third parties are protected.

VSX, Vogel Software, Spaix, Impeller.net, BProX und PipeCalc are registered trademarks or trademarks of VSX – VOGEL SOFTWARE GmbH in the European Union, USA and other countries.

Conditions of use:

The software described in this manual will be provided according to the conditions of the licensing agreement and may be only used under the described conditions.

The current software licensing conditions can be found on:
http://www.vsx.net/docs/software_en.pdf