SAPlugin Manual – v1.3

Software Architecture 2016 – 2017

Abstract

This manual briefly explains how you can use the `SAPlugin` to automatically export your element catalog from *Visual Paradigm* to LaTeX. The `SAPlugin` reads the descriptions and other metadata in your *Visual Paradigm* model and automatically exports the *Element Catalog* section of your part 2 report to LaTeX, in the required template.

1 Set-up

This section covers the plug-in requirements and its installation.

1.1 Plug-in requirements

The `SAPlugin` requires the following Visual Paradigm version to run: Visual Paradigm 14.0 build 20170420*ag*[1] (or later). This version is installed in the PC classes. Make sure you run version 14, and not 13 which is also available.

Use `Help` ▷ `Update`, and then choose `Update to latest patch` at the bottom left side, to upgrade to the latest build.

1.2 Plug-in installation

Use `Help` ▷ `Install Plugin` and select the provided zip file:

`be.kuleuven.cs.distrinet.sa.vpplugin.zip`

Restart Visual Paradigm to complete the installation.

Changed in v1.1

You can install new versions of the plugin over the previous version. There is no need to uninstall the previous version.

Changed in v1.3

When installing an upgrade of `SAPlugin` make sure the *zip*-file has the same name as specified above. *Visual Paradigm* uses this name to install the plugin. If the name differs, *Visual Paradigm* will try to install the new version next to the old one and complain that the pluginID of the new version is already in use.

2 Structure of your Visual Paradigm project

Automatically exporting your element catalog from *Visual Paradigm* requires you to maintain a sound model of your architecture. This section elaborates on how `SAPlugin` reads and interprets your model in order to be able to properly export it.

[1] You can check the build number in the about screen

2.1 Exporting Component Descriptions

The super- and sub-component relations need to be explicitly in your model. Verify that, in your model explorer, components are correctly nested corresponding to their super- or sub-component relationships.

→ Subsection 2.2

This export functionality automatically includes a reference to all interfaces that are required or provided by a component.

! →

If you duplicate interfaces on different diagrams, they will be listed multiple times here. This happens because, according to your model, your component provides several interfaces with the same name. To avoid this, drag the existing interface from the model explorer to the other diagrams (to create an auxiliary view).

Responsibility

The description of the responsibilities of a component are extracted from the component's description (on the *General* tab of its specification). All names of known components that appear in this description (case-sensitive) will be replaced automatically by references to that component.

2.2 Exporting Interfaces and their Operations

When exporting interfaces, the `SAPlugin` will create a list of all the interfaces in your project, and for each interface, list the operations of that interface.

To properly extract the operations, it is necessary for your project to define all operations for each interface.

! →

Make sure that you use valid types for the parameter and return types in your operations. Every type used in your operations should be a valid Data Type.

→ subsection 2.4

Changed in v1.3

If an invalid type is specified, the exported catalog will show: Invalid type: [Type (e.g., com.vp.plugin.model.IComponent)] ElementName.

Effect

Changed in v1.2

The effect of an operation is extracted from the operations description (on the *General* tab of its specification).

2.3 Exporting Exceptions

Visual Paradigm does not have a separate model element for representing exceptions. `SAPlugin` automatically determines if a class is an exception, when it is listed in the *Raised Exceptions* tab of an operation. When the exception is listed there, it will also be automatically included in the *throws* clause of that operation's signature.

! →

Exception classes that are not raised anywhere will be listed between the Data Types.

2.4 Exporting Data Types

Make a class for all data types you use. These will be collected by the `SAPlugin` and listed in the *DataTypes* section of the element catalog.

Primitive types

A number of primitives are supported by default. You can use these anywhere you can use your own data types. There are the following:

- boolean
- byte
- char
- double
- float
- int
- long
- short
- string
- void

Container types — The following container types are recognized by the `SAPlugin`, so you do not need to define a class for them:

- List
- Map
- Set
- Tuple

New in v1.1 — Use generics (`< >`) to specify the types they contain, just as you would in Java. `SAPlugin` will recognize these types as well. Furthermore, you can also use any self-defined class, if desired.

Changed in v1.1 — Previously defined container types with square brackets (`[ ]`) will continue to work as before. But, you may experience issues when using them for entering the return type or attribute type directly on a class diagram.[2]

! → These container types are only recognized if you also specify the type they contain.

! → `SAPlugin` does not check the number of type arguments to include for each container type (i.e., one for List and Set, and two for Map).

Example — `Map<SomeClass,boolean>`

Description — The description of a data type is extracted from the data type's description (on the *General* tab of its specification.)

2.4.1 Data type attributes

New in v1.1 — You can specify attributes of your data types. These will also be included in the export. Limit yourself to specifying attributes that are relevant with regard to realizing your quality requirements.

! → The types of these attributes need to exist as well

2.5 Missing elements

DataTypes — Missing elements such as non-existing data types will be output by the tool as follows: UndefinedDataType.

Descriptions — Missing descriptions will be marked by the tool as Undefined.

! → Make sure to check your final report for missing elements.

2.6 Descriptions and responsibilities

Changed in v1.2 — All names of known components that appear in descriptions (case-sensitive) will be replaced automatically by references to that component.

New in v1.2 — It is possible that you want to describe multiple components and still include a reference. If you use the plural form 'as-is' the name will not match and will not be replaced by a reference. If you do want it to be replaced by a reference, use '&' to append the suffix. E.g., Gateway&s will match and will be replaced with a reference and the suffix: `Gateways`.

[2] Visual Paradigm interprets this as a type modifier

! → *Avoid having multiple components with the same name.*

New in v1.3

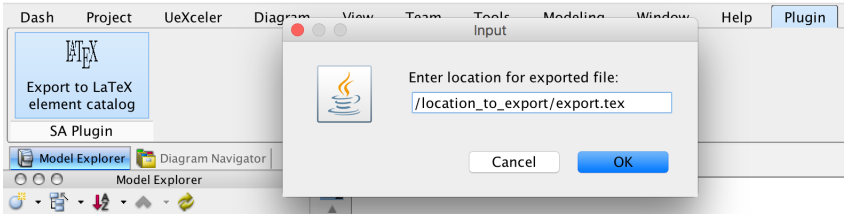
If you have multiple components with the same name, the plugin cannot replace their names with references in the export. If you do want to add references, you will have to use the fully qualified names of components (e.g., `Components.SomeComponent.SomeSubComponent`) everywhere.

3 Using the Plug-in to Export the Catalog to LaTeX

3.1 Visual Paradigm UI

You can use the plug-in via the plug-in tab in Visual Paradigm. Go to *Plugin* ▷ *Export catalog to LaTeX*. Next, you will be asked to provide the location of the output `.tex` file.

Figure 1: Screenshot of the export to LaTeX element catalog.



! → Make sure there is no other file at that location, the plug-in will overwrite that file without warning!

3.2 Command Line Interface

Changed in v1.3

You can also run the plugin via the command line to automatically export your catalog to LaTeX without manual intervention. Below are the currently supported command line arguments:

```
usage: Plugin.[sh|bat] -project "/path/to/project.vpp"
         -pluginid "SAPlugin" -pluginargs "'[-about]
         [-e exportfile] [-h] [-v] [-version]'"
-about               About SAPlugin.
-e,--export <arg>    LaTeX export to the specified
                     destination-file.
-h,--help            Print this help message.
-v,--verbose         Output additional information.
-version             Show SAPlugin version.
```

! → Note the combination of quotes `"'"`; this is necessary to prevent *Visual Paradigm* from (mis)interpreting the arguments.

Plugin.[sh|bat] This script is provided by *Visual Paradigm* in the *scripts* folder in the installation directory. You will need to use this script to execute *Visual Paradigm* via the command line and forward the commands to `SAPlugin`. For more information on locating and using this script, we refer you to the *Visual Paradigm* website.[3]

Known errors We are aware that running the plugin from the command line will output several `NullPointerExceptions` from the *AWT-EventQueue-0*-thread to the console. This is an issue in *Visual Paradigm* not the plugin.
→ Section 5
If you encounter other exceptions, you believe are caused by the plugin, let us know of course.

[3] `https://www.visual-paradigm.com/support/documents/vpuserguide/124/255/84449_runningplug-.html`

4 Exported Catalog

Changed in v1.3

The exported catalog requires the following LaTeX packages:

```
\usepackage{enumitem}
\usepackage{tikz}
\usepackage{nameref}
\usepackage{hyperref}
```

All of these packages are already loaded in the *SAReport.cls* template that has been provided to you.

5 Feedback and issues

If you run into a problem with `SAPlugin`, or if you have a feature request or other feedback to share with us, contact us at:

`SoftwareArch2017@cs.kuleuven.be`

Changed in v1.3

In case of a problem, include at least the following information:

- Description of the problem and steps to reproduce. What were you doing, and what went wrong?
- Your *Visual Paradigm* project (or a minimal example) which allows us to reproduce the problem
- *Visual Paradigm* version and build number
- `SAPlugin` version
- Log file with the exceptions (`About` ▷ `Export Log File`).

6 Version History

6.1 Version 1.1

Added in version 1.1:

- Added error message if an operation contains parameter or return types that are not data types.
- Added full support for generics (`< >`) specification as in Java.
- Attributes of data types, if present, are exported as well.
- Notification if a new version of the plugin is available.

Changed in version 1.1:

- Clarified explanation on installing new versions of the plugin.
- Clarified explanation on square brackets for container types.

6.2 Version 1.2

Added in version 1.2:

- Added support for running `SAPlugin` from the command line.
- Added support for referring to plurals of components using '&'.

Changed in version 1.2:

- Clarified explanation on replacing components in the description by a reference in LaTeX.

6.3 Version 1.3

Added in version 1.3:

- Fixed issue in parsing references when there are duplicate components.
- Support for referring to components using their fully qualified name.

Changed in version 1.3:

- Clarified `SAPlugin` upgrade instructions on zip naming issue.
- Changed behaviour for invalid types: the plugin will always export but mark them as invalid.
- Clarified LaTeX dependencies of exported catalog.
- Version and about CLI commands.
- Provide `SAPlugin` version number as well when communicating feedback and issues.