BMW GROUP

Rolls-Royce Motor Cars Limited

User Guide PAGe

Project	BMW AUTOSAR Core 4 Rel. 3 and adaptive BMW AUTOSAR Core Rel. 1
Author	BMW AG
Release Date	2017-11-09
Version	1.1.0
Status	Release
Hotline	+49 89 382 - 32233 (classic) / +49 89 382 - 22522 (adaptive)
Contact	bac@bmw.de (classic) / abac@bmw.de (adaptive) https://asc.bmw.com/jira/browse/BSUP (extern) https://asc.bmwgroup.net/jira/browse/BSUP (intern)

Revision History

Version	Date	Changed by	Description
1.1.0	2017-11-09	JC-42	Initial Documentation Release

Company
Bayerische Motoren Werke Aktiengesellschaft
Postal address
BMW AG
80788 München
Office address
Forschungs- und Innovationszentrum (FIZ)
Hufelandstr. 1
80937 München
Telephone
Switchboard
+49 89 382-0
Internet
www.bmwgroup.com

Table of Contents

1 Overview

Purpose

This user guide describes the functionality and usage of the Python AUTOSAR Generator (PAGe).

PAGe is a generator for text templates which has some knowledge of the AUTOSAR Meta Model regarding parameter configuration. It allows easy access of modules, containers and values as well as providing loops and verify a paramconf against a paramdef. These options are also available for postbuild.

PAGe is based on Python 3. It uses only modules of the standard library.

PAGe is designed for the usage with the BMW AUTOSAR Core modules.

Input files must have the ending `.pgen` which is removed from the output filename.

Usage

Generation

To generate a pgen file.

```
python3 -m page example.pgen
```

If arxmls should be used pass them to the commandline as well.

```
python3 -m page -o /tmp example.pgen input.arxml
```

Also multiple pgen files and arxml files work.

```
python3 -m page -o /tmp example1.pgen example2.pgen input.arxml input_new.arxml
```

Validation

In case you want to check a paramconf against its corresponding paramdef you can use the following commandline `python3 -m page paramconf.arxml paramdef.arxml`

The validation will be performed for all the variants if an ecuc configuration is provided, too.

```
python3 -m page paramconf.arxml paramdef.arxml ecuc.arxml
```

verb

To enable a more verbose output add `-v` to `-vvvv` to the command line.

2 Acronyms and Abbreviations

API	Application Programming Interface
Application	Application stands for the high-level part of software that uses the APIs provided by the modules. It can also mean the driving application that does not belong to the Bootloader.
AUTOSAR	Automotive Open System Architecture
OS	Operating System
PAGe	Python AUTOSAR Generator
pgen	Page generation file

3 Functionality

PAGe builds a model of the provided ARXML paramconf and paramdefs. Within the pgen files you have access to this model and can navigate and retrieve values easily. Besides the access to the data you have the full power of python.

Command Line Usage

Either you install page or you add the main directory to the source path. Installation can be achieved by

```
python3 setup.py install
```

after that the `page` command will be available. So you can call PAGe using:

```
page input.pgen test.arxml -vvvv
```

But you may want to do install page only in a virtualenv so it does not harm any other installation.

-o Specify the output directory, default is the current directory.
--stdout do not write files, print output to stdout instead.
-vvv change level of verbosity.
-l Location to write the logfile to. Default is stdout.
-m Specify delimiter to be used in pgen files. This allows you to specify other delimiters than the default e.g. you have files, where the defaults have a special meaning and you don't like to escape the every time used.
-d Enable debug mode (trap functions become active)
-V Verifies given paramconf against the paramdefs which ara supplied.
-i Add paths for the search of includes.
-D Pass variables to the pgen files (`params` dictionary)
-O Change the optimisedb. This database stores the hash and the filename, so file with no change will not be overwritten.
-f Forces write of files, even if no output change has been detected.

Fileformat

The pgen files shall be utf-8 encoded files. The files shall use unix file endings, windows file endings `\r\n` are transformed to `\n`.

A file consists of text and several blocks that are interpreted by PAGe. These blocks are enclosed by a special sequence of characters. `%{` marks the beginning of a block and `}%` marks the end. The first character occuring directly after the opening tag has a special meaning an defines the type of the block. The following distinct blocks are supported:

= The result of the statement within the tags are printed as output. A single statement shall be used but can contain newline and indentiation. The result of the statement will appear in the output file.
This marker indicates a section of code. The code can consist of any valid piece of python code and is also able to use the access functions for the data of the ARXMLs. It can be indentiated and consist

of multiple lines. For indentiation the first line is taken for reference, the rest of the indentiation has to follow Pythons rules.

: For conditions the colon is used as a marker. It occurs at least two times, one for the condition and one time (without extra text) for the closure. This is needed for the correct handling. You still have to use `if`, `elif` and `else` keywords.

? For quick conditionals you can use a question mark. It works also directly as output. The condition followed by a colon which seperates it from the part which is written in case of success. And an optional second colon to seperate the ouput in the case the condition is not fullfilled.

@ This one loops over a statement. You can loop over a number of container instances, or any other python iterateable. To iterate over a python iterateable use the syntax `myitem in mycontainer` which makes the current item available as `myitem`. For loop over arxml elements, please have a look at **??**

+ Use the plus sign for including other pgen or python files. These files are included and interpreted at the current point within the file which they are included. A file can be included several times. Pure python files can be included, too. By default the current path of the currently leaded source files (or included files) is used as search directory. You can add directories using the command line switch `-i`

~ This mark is for special handling of post build variation. It loops over the configured pre defined variants that are configured for the ecu. Within the context of this mark, you have two special variables available: `predefined_variant_name` which is the name of the current variant or `None` if no variantation is configured. And `predefined_variant_postfix` which contains an underscore followed by the variant name or an empty string else. This can be used to build the name of the configuration according to ECUConfiguration.

Available Function and Objects

The following functions are provided as part of PAGe:

comment Format a given string as a comment in the current context. If no delimiter are given, the functions calls the automatic delimiter resolution. The string is splitted into single lines and enclosed in the delimiter
set_comment_delimiter Set the comment markers
count Count elements found by shortname/search expression.
current_file Get the current filename of the file processed (pgen file).
current_shortname Get the AUTOSAR shortname of the current scope.
current_shortname_path Get the AUTOSAR shortname path.
each Get unique shortname paths of the elements that match the given search expression.
exists Checks existence of elements specified by search expression or shortname in current context.
generation_file_fullpath Full path of the source file.
generation_file_modification Date of modification of the source file.
generation_info Info about the generation as dict.
generation_timestamp Timestamp of generation.
generation_tool name of the generation tool.
generation_version_info Version of Generator.
get_elements Get list of elements matching search expression.
get_xpaths Deprecated for get_elements.
into Change context to given target, resets if target is None. The scope is held in a stack
join Join a set of values referenced by the search expression.
leave Leave the current scope.
reset Resets the context to the root node.

set_debug Enable pdb debugging.
shortname Get the shortname at the given shortname path expression,
trap Traps if debugging is enabled.
value Fetching the value or value-ref of a search expression. Values are automatically casted to the correct type.
write Write arguments to the output.
container Selects containers that refere to a given definition ref.
module Selects modules that refere to a given definition ref.
ref Get the elements in the model, which refere to the given shortnamepath as definition ref.
snpath Get the element at the given shortname path.

Additionally to the functions and objects there is a dictionary called `params` which contains parameters passed to page by the command line (`-D test=hello`).

Handling of paramconf

In general when using the data of a paramconf you should never enter a specific path you expect to be in the configuration. You should only use paths you define in the parameter definition or that are contained in the configuration as value (value refs).

Notes ARXML shortcuts

The shortnamepaths `snpath` are looked up in the following order:

. return element
.. return parent
contains * Search children
startwith / lookup full path from root node
else lookup relative path

This means, whenever you want to get a grandchild of the current scope you need to prepend a star to the node you are looking for or use the search.

Loops

In case you loop over VALUE or VALUE-REF elements, you can access its value using `value()` without a target specified.

4 Examples

Basic Navigation

This example will demonstrate basic navigation through an ARXML document.

The example contains a module named `MyModule` which is of type `TestModule` and located in the package `/TestPackage/TestModule`

It will demonstrate the following commands:

- into
- reset
- ref
- module
- current_shortname_path
- with

ARXML Files

input.arxml

```
<?xml version='1.0'?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0␣autosar.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>TestPackage</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>TestModule</SHORT-NAME>
          <ELEMENTS>
            <ECUC-MODULE-CONFIGURATION-VALUES>
              <SHORT-NAME>MyModule</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/TestModule</DEFINITION-REF>
              <CONTAINERS>
                <ECUC-CONTAINER-VALUE>
                  <SHORT-NAME>NumericalValues</SHORT-NAME>
                  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/TestModule/
                     NumericalValues</DEFINITION-REF>
                  <PARAMETER-VALUES>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                         NumericalValues/ValueInt</DEFINITION-REF>
                      <VALUE>5</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                         NumericalValues/ValueInt</DEFINITION-REF>
                      <VALUE>8</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                    <ECUC-NUMERICAL-PARAM-VALUE>
```

```
                    <DEFINITION-REF DEST="ECUC-FLOAT-PARAM-DEF">/BMW_DEF/TestModule/
                       NumericalValues/ValueFloat</DEFINITION-REF>
                    <VALUE>3.1415</VALUE>
                  </ECUC-NUMERICAL-PARAM-VALUE>
                </PARAMETER-VALUES>
              </ECUC-CONTAINER-VALUE>
            </CONTAINERS>
          </ECUC-MODULE-CONFIGURATION-VALUES>
          <ECUC-MODULE-CONFIGURATION-VALUES>
            <SHORT-NAME>NotMyModule</SHORT-NAME>
            <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/AnotherTestModule</DEFINITION-
               REF>
            <CONTAINERS>
              <ECUC-CONTAINER-VALUE>
                <SHORT-NAME>StringValues</SHORT-NAME>
                <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/
                   AnotherTestModule/StringValues</DEFINITION-REF>
                <PARAMETER-VALUES>
                  <ECUC-NUMERICAL-PARAM-VALUE>
                    <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/BMW_DEF/
                       AnotherTestModule/StringValues/ValueString</DEFINITION-REF>
                    <VALUE>Hello World the value is: </VALUE>
                  </ECUC-NUMERICAL-PARAM-VALUE>
                </PARAMETER-VALUES>
              </ECUC-CONTAINER-VALUE>
            </CONTAINERS>
          </ECUC-MODULE-CONFIGURATION-VALUES>
        </ELEMENTS>
      </AR-PACKAGE>
    </AR-PACKAGES>
  </AR-PACKAGE>
 </AR-PACKAGES>
</AUTOSAR>
```

pgen Files

input.pgen

```
%{# into(ref('TestModule')) }%
%{= current_shortname_path() }%
This will change into the *FIRST* element found
%{# into(ref('ValueInt')) }%
The shortname will be taken from the parent element,
  as this element is not an identifyable.
%{= current_shortname_path() }%
%{= value() }%
%{# leave() }%
If you expect more than one use a loop
For the loop the context is automatically changed
%{@ ref('ValueInt') }%
%{= value() }%
%{@}%
%{# into(ref('ValueFloat')) }%
%{= current_shortname_path() }%
%{= value() }%
%{# reset() }%
%{= current_shortname_path() }%
```

```
%{# into(ref('TestModule')) }%
%{= current_shortname_path() }%
%{# into() }%
%{= current_shortname_path() }%
For a short interaction with a certain element within a code block,
  you can also use the with statement of python
%{#
  with module('TestModule'):
    input = value(ref('ValueFloat'))
  input = input * 2
  with module('AnotherTestModule'):
    string = value(ref('ValueString'))
}%
%{= '{} {}'.format(string, input) }%
```

Command line

```
page -vvvv input.arxml input.pgen
```

Console Output

```
BMW PAGe    : INFO     Reading XML: input.arxml
BMW PAGe    : INFO     Reading XML took: 0.000859 s
BMW PAGe    : INFO     Reading all XMLs took: 0.00104 s
BMW PAGe    : INFO     processing Pgen: input.pgen
BMW PAGe    : INFO     Open file /001/input.pgen
BMW PAGe    : INFO     Processing Pgen took: 0.00317 s
BMW PAGe    : INFO     File does not exist
BMW PAGe    : INFO     Write /001/input
```

File Output

```
/TestPackage/TestModule/MyModule
This will change into the *FIRST* element found
The shortname will be taken from the parent element,
  as this element is not an identifyable.
/TestPackage/TestModule/MyModule/NumericalValues
5
If you expect more than one use a loop
For the loop the context is automatically changed
5
8
/TestPackage/TestModule/MyModule/NumericalValues
3.1415
ROOT
/TestPackage/TestModule/MyModule
ROOT
For a short interaction with a certain element within a code block,
  you can also use the with statement of python
Hello World the value is:  6.283
```

Logging

This example will demonstrate the logging functionality.

There are several log levels that depend on the -vs provided on the command line.

It will demonstrate the following commands:

- logger.debug
- logger.info
- logger.warning
- logger.error
- logger.fatal

pgen Files

input.pgen

```
Logging of some information

%{# logger.debug('This is a message for debugging so you have to specify a few v
%{# logger.info('This text will appear in the INFO log') }%
%{# logger.warning('You''ve been warned')}%
%{# logger.error('This will not work, ok I\'ll let you continue')}%
%{# logger.fatal('Oh noooo!')}%
```

Command line

```
page -vvvv input.pgen
```

Console Output

```
BMW PAGe    : INFO     processing Pgen: input.pgen
BMW PAGe    : INFO     Open file /002/input.pgen
BMW PAGe    : DEBUG    This is a message for debugging so you have to specify a
BMW PAGe    : INFO     This text will appear in the INFO log
BMW PAGe    : WARNING  Youve been warned
BMW PAGe    : ERROR    This will not work, ok I'll let you continue
BMW PAGe    : CRITICAL Oh noooo!
BMW PAGe    : INFO     Processing Pgen took: 0.00185 s
BMW PAGe    : INFO     File does not exist
BMW PAGe    : INFO     Write /002/input
```

File Output

```
Logging of some information
```

Usage of Code Blocks

This example will show a few details about the code block

The code block is one of PAGes super power. It allows to use any python code from within it. You can still use PAGes functions but also any other python statement you could think of.

It will demonstrate the following commands:

- write
- CodeBlock

pgen Files

input.pgen

```
Usage of Code Blocks

Besides the output blocks you can use code blocks:
%{#
  # The first line defines the basic indentiation.
  # all lines are interpreted as pure python.
  def MyFunction(a, b):
    return (a+b)*b
}%

All variables, functions, ect. will remain until the end of the pgen file.

You can now see the result of the previous function: %{= MyFunction(3, 5) }%

If you want to write to the output from within a code block, you should use the
%{#
  for x in range(1, 15):
    if '3' in str(x) or x % 3 == 0:
      write('buzz\n')
    else:
      write('{}\n'.format(x))
}%
NB: write does not automatically add a newline.
```

Command line

```
page –vvvv input.pgen
```

Console Output

```
BMW PAGe    : INFO     processing Pgen: input.pgen
BMW PAGe    : INFO     Open file /003/input.pgen
BMW PAGe    : INFO     Processing Pgen took: 0.00171 s
BMW PAGe    : INFO     File does not exist
BMW PAGe    : INFO     Write /003/input
```

File Output

Usage of Code Blocks

Besides the output blocks you can use code blocks:

All variables, functions, ect. will remain until the end of the pgen file.

You can now see the result of the previous function: 40

If you want to write to the output from within a code block, you should use the
1
2
buzz
4
5
buzz
7
8
buzz
10
11
buzz
buzz
14

NB: write does not automatically add a newline.

Conditionals

This example will show the usage of conditional parts within a pgen file

The example contains a module named `MyModule` which is of type `TestModule` and located in the package `/TestPackage/TestModule`

It will demonstrate the following commands:

- if
- elif
- else
- ShortIfBlock
- ConditionalBlock
- value

ARXML Files

input.arxml

```
<?xml version='1.0'?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="http://autosar.org/schema/r4.0␣autosar.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>TestPackage</SHORT-NAME>
      <ELEMENTS>
        <ECUC-MODULE-CONFIGURATION-VALUES>
          <SHORT-NAME>MyModule</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/TestModule</DEFINITION-REF>
          <CONTAINERS>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>NumericalValues</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/TestModule/
                  NumericalValues</DEFINITION-REF>
              <PARAMETER-VALUES>
                <ECUC-NUMERICAL-PARAM-VALUE>
                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                      NumericalValues/ValueInt</DEFINITION-REF>
                  <VALUE>5</VALUE>
                </ECUC-NUMERICAL-PARAM-VALUE>
              </PARAMETER-VALUES>
            </ECUC-CONTAINER-VALUE>
          </CONTAINERS>
        </ECUC-MODULE-CONFIGURATION-VALUES>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

pgen Files

input.pgen

```
Conditionals can be done as seen here:
%{: if value(ref('ValueInt')) < 6 }%
Test
%{: elif exists(ref('ValueFloat')) }%
This will not occur in the output
%{: else }%
This won't appear
%{:}%

Want some shorter if?
These write the text provided dependent on the result of the statement evaluation
# Both options provided
True:  %{? True :For sure!:Nooooo!}%
False: %{? False:For sure!:Nooooo!}%
# No is not an option
True:  %{? True :For sure!}%
False: %{? False:For sure!}%
# Only output something if the result is false
True:  %{? True ::NO!!!!}%
False: %{? False::NO!!!!}%
```

Command line

```
page –vvvv input.arxml input.pgen
```

Console Output

```
BMW PAGe     : INFO     Reading XML: input.arxml
BMW PAGe     : INFO     Reading XML took: 0.000677 s
BMW PAGe     : INFO     Reading all XMLs took: 0.000871 s
BMW PAGe     : INFO     processing Pgen: input.pgen
BMW PAGe     : INFO     Open file /004/input.pgen
BMW PAGe     : INFO     Processing Pgen took: 0.00211 s
BMW PAGe     : INFO     File does not exist
BMW PAGe     : INFO     Write /004/input
```

File Output

```
Conditionals can be done as seen here:
Test

Want some shorter if?
These write the text provided dependent on the result of the statement evaluation
# Both options provided
True:  For sure!
False: Nooooo!
# No is not an option
True:  For sure!
False:
# Only output something if the result is false
True:
False: NO!!!!
```

More Functions

This example will contain usages of different functions.

The example contains a module named `MyModule` which is of type `TestModule` and located in the package `/TestPackage/TestModule`

It will demonstrate the following commands:

- count
- current_shortname
- exists
- with

ARXML Files

input.arxml

```
<?xml version='1.0'?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0 autosar.xsd">
```

```
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>TestPackage</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>TestModule</SHORT-NAME>
          <ELEMENTS>
            <ECUC-MODULE-CONFIGURATION-VALUES>
              <SHORT-NAME>MyModule</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/TestModule</DEFINITION-REF>
              <CONTAINERS>
                <ECUC-CONTAINER-VALUE>
                  <SHORT-NAME>NumericalValues</SHORT-NAME>
                  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/TestModule/
                     NumericalValues</DEFINITION-REF>
                  <PARAMETER-VALUES>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                         NumericalValues/ValueInt</DEFINITION-REF>
                      <VALUE>5</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                         NumericalValues/ValueInt</DEFINITION-REF>
                      <VALUE>8</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-FLOAT-PARAM-DEF">/BMW_DEF/TestModule/
                         NumericalValues/ValueFloat</DEFINITION-REF>
                      <VALUE>3.1415</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                  </PARAMETER-VALUES>
                </ECUC-CONTAINER-VALUE>
              </CONTAINERS>
            </ECUC-MODULE-CONFIGURATION-VALUES>
            <ECUC-MODULE-CONFIGURATION-VALUES>
              <SHORT-NAME>NotMyModule</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/AnotherTestModule</DEFINITION-
                 REF>
              <CONTAINERS>
                <ECUC-CONTAINER-VALUE>
                  <SHORT-NAME>StringValues</SHORT-NAME>
                  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/
                     AnotherTestModule/StringValues</DEFINITION-REF>
                  <PARAMETER-VALUES>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/BMW_DEF/
                         AnotherTestModule/StringValues/ValueString</DEFINITION-REF>
                      <VALUE>Hello World the value is: </VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                  </PARAMETER-VALUES>
                </ECUC-CONTAINER-VALUE>
              </CONTAINERS>
            </ECUC-MODULE-CONFIGURATION-VALUES>
          </ELEMENTS>
        </AR-PACKAGE>
      </AR-PACKAGES>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

pgen Files

input.pgen

```
%{# into(ref('TestModule')) }%
%{= current_shortname_path() }%
We can also check the number of elements:
There are %{= count(ref('ValueInt')) }% ValueInt elements.
There exists at least one ValueFloat: %{= exists(ref('ValueFloat')) }%.

You can use the previous options also as an asignment statement:
%{# previous_sn = current_shortname() }%
%{# leave()}%
and use it later
%{= previous_sn }%
```

Command line

```
page –vvvv input.arxml input.pgen
```

Console Output

```
BMW PAGe    : INFO     Reading XML: input.arxml
BMW PAGe    : INFO     Reading XML took: 0.0012 s
BMW PAGe    : INFO     Reading all XMLs took: 0.00162 s
BMW PAGe    : INFO     processing Pgen: input.pgen
BMW PAGe    : INFO     Open file /005/input.pgen
BMW PAGe    : INFO     Processing Pgen took: 0.00212 s
BMW PAGe    : INFO     File does not exist
BMW PAGe    : INFO     Write /005/input
```

File Output

```
/TestPackage/TestModule/MyModule
We can also check the number of elements:
There are 2 ValueInt elements.
There exists at least one ValueFloat: 1.

You can use the previous options also as an asignment statement:
and use it later
MyModule
```

ARXML Stuff

This example will demonstrate some more ARXML selection stuff which can be usefull.

It will demonstrate the following commands:

- into
- module

- each
- container
- ref
- snpath

ARXML Files

input.arxml

```
<?xml version='1.0'?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0␣autosar.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>TestPackage</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>TestModule</SHORT-NAME>
          <ELEMENTS>
            <ECUC-MODULE-CONFIGURATION-VALUES>
              <SHORT-NAME>MyModule</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/TestModule</DEFINITION-REF>
              <CONTAINERS>
                <ECUC-CONTAINER-VALUE>
                  <SHORT-NAME>NumericalValues</SHORT-NAME>
                  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/TestModule/
                      NumericalValues</DEFINITION-REF>
                  <PARAMETER-VALUES>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                          NumericalValues/ValueInt</DEFINITION-REF>
                      <VALUE>5</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                          NumericalValues/ValueInt</DEFINITION-REF>
                      <VALUE>8</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-FLOAT-PARAM-DEF">/BMW_DEF/TestModule/
                          NumericalValues/ValueFloat</DEFINITION-REF>
                      <VALUE>3.1415</VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                  </PARAMETER-VALUES>
                </ECUC-CONTAINER-VALUE>
              </CONTAINERS>
            </ECUC-MODULE-CONFIGURATION-VALUES>
            <ECUC-MODULE-CONFIGURATION-VALUES>
              <SHORT-NAME>NotMyModule</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/AnotherTestModule</DEFINITION-
                  REF>
              <CONTAINERS>
                <ECUC-CONTAINER-VALUE>
                  <SHORT-NAME>StringValues</SHORT-NAME>
                  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/
                      AnotherTestModule/StringValues</DEFINITION-REF>
                  <PARAMETER-VALUES>
```

```
                    <ECUC-NUMERICAL-PARAM-VALUE>
                      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/BMW_DEF/
                         AnotherTestModule/StringValues/ValueString</DEFINITION-REF>
                      <VALUE>Hello World the value is: </VALUE>
                    </ECUC-NUMERICAL-PARAM-VALUE>
                  </PARAMETER-VALUES>
                  <REFERENCE-VALUES>
                    <ECUC-REFERENCE-VALUE>
                      <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/BMW_DEF/AnotherTestModule/
                         StringValues/ReferenceToNumericals</DEFINITION-REF>
                      <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/TestPackage/TestModule/
                         MyModule/NumericalValues</VALUE-REF>
                    </ECUC-REFERENCE-VALUE>
                  </REFERENCE-VALUES>
                </ECUC-CONTAINER-VALUE>
              </CONTAINERS>
            </ECUC-MODULE-CONFIGURATION-VALUES>
          </ELEMENTS>
        </AR-PACKAGE>
      </AR-PACKAGES>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

pgen Files

input.pgen

```
Find a Module and change the current scope to it
%{# into( module('TestModule')) }%
The next element to get to is a (sub)container
%{# into( container('NumericalValues')) }%
But we need a different container
%{#
  reset()
  into(container('AnotherTestModule/StringValues'))
}%
You can search for elements references, too:
%{= value( ref('ReferenceToNumericals')) }%
You can use these to go into but you have to use value
to get the string of the reference
%{# into(value(ref('ReferenceToNumericals'))) }%
%{= current_shortname_path() }%
%{# reset() }%
```

Command line

```
page -vvvv input.arxml input.pgen
```

Console Output

```
BMW PAGe    : INFO     Reading XML: input.arxml
BMW PAGe    : INFO     Reading XML took: 0.00095 s
BMW PAGe    : INFO     Reading all XMLs took: 0.00114 s
BMW PAGe    : INFO     processing Pgen: input.pgen
```

```
BMW PAGe    : INFO     Open file /006/input.pgen
BMW PAGe    : INFO     Processing Pgen took: 0.00228 s
BMW PAGe    : INFO     File does not exist
BMW PAGe    : INFO     Write /006/input
```

File Output

```
Find a Module and change the current scope to it
The next element to get to is a (sub)container
But we need a different container
You can search for elements references, too:
/TestPackage/TestModule/MyModule/NumericalValues
You can use these to go into but you have to use value
to get the string of the reference
/TestPackage/TestModule/MyModule/NumericalValues
```

Filtering Elements

This example will demonstrate some techniques that can be used to filter some elements out of a larger list.

It will demonstrate the following commands:

- into
- reset
- ref
- module
- current_shortname_path
- with

ARXML Files

input.arxml

```
<?xml version='1.0'?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0␣autosar.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>TestPackage</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>TestModule</SHORT-NAME>
          <ELEMENTS>
            <ECUC-MODULE-CONFIGURATION-VALUES>
              <SHORT-NAME>MyModule</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/TestModule</DEFINITION-REF>
              <CONTAINERS>
                <ECUC-CONTAINER-VALUE>
                  <SHORT-NAME>NumericalValues</SHORT-NAME>
                  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/TestModule/
                      NumericalValues</DEFINITION-REF>
```

```
                <PARAMETER-VALUES>
                  <ECUC-NUMERICAL-PARAM-VALUE>
                    <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                       NumericalValues/ValueInt</DEFINITION-REF>
                    <VALUE>5</VALUE>
                  </ECUC-NUMERICAL-PARAM-VALUE>
                  <ECUC-NUMERICAL-PARAM-VALUE>
                    <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/BMW_DEF/TestModule/
                       NumericalValues/ValueInt</DEFINITION-REF>
                    <VALUE>8</VALUE>
                  </ECUC-NUMERICAL-PARAM-VALUE>
                  <ECUC-NUMERICAL-PARAM-VALUE>
                    <DEFINITION-REF DEST="ECUC-FLOAT-PARAM-DEF">/BMW_DEF/TestModule/
                       NumericalValues/ValueFloat</DEFINITION-REF>
                    <VALUE>3.1415</VALUE>
                  </ECUC-NUMERICAL-PARAM-VALUE>
                </PARAMETER-VALUES>
              </ECUC-CONTAINER-VALUE>
            </CONTAINERS>
          </ECUC-MODULE-CONFIGURATION-VALUES>
          <ECUC-MODULE-CONFIGURATION-VALUES>
            <SHORT-NAME>NotMyModule</SHORT-NAME>
            <DEFINITION-REF DEST="ECUC-MODULE-DEF">/BMW_DEF/AnotherTestModule</DEFINITION-
               REF>
            <CONTAINERS>
              <ECUC-CONTAINER-VALUE>
                <SHORT-NAME>StringValues</SHORT-NAME>
                <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/BMW_DEF/
                   AnotherTestModule/StringValues</DEFINITION-REF>
                <PARAMETER-VALUES>
                  <ECUC-NUMERICAL-PARAM-VALUE>
                    <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/BMW_DEF/
                       AnotherTestModule/StringValues/ValueString</DEFINITION-REF>
                    <VALUE>Hello World the value is: </VALUE>
                  </ECUC-NUMERICAL-PARAM-VALUE>
                </PARAMETER-VALUES>
              </ECUC-CONTAINER-VALUE>
            </CONTAINERS>
          </ECUC-MODULE-CONFIGURATION-VALUES>
        </ELEMENTS>
      </AR-PACKAGE>
    </AR-PACKAGES>
  </AR-PACKAGE>
 </AR-PACKAGES>
</AUTOSAR>
```

pgen Files

input.pgen

Command line

```
page -vvvv input.arxml input.pgen
```

Console Output

```
BMW PAGe    : INFO      Reading XML: input.arxml
BMW PAGe    : INFO      Reading XML took: 0.000903 s
BMW PAGe    : INFO      Reading all XMLs took: 0.00111 s
```

```
BMW PAGe    : INFO     processing Pgen: input.pgen
BMW PAGe    : INFO     Open file /007/input.pgen
BMW PAGe    : INFO     Processing Pgen took: 0.00141 s
BMW PAGe    : INFO     Empty content -> no file written
BMW PAGe    : INFO     Skipping write of /007/input, no changes detected
```

File Output