Control Commands

Table of contents

1. Control Commands

1.1. Include

The purpose of this command is to include another instruction file at exactly the point where this command is invoked. The *Include* command cannot be put inside a *Data Source Adapter*. It can only be used directly under *Instructions* or one of the following control commands if they are directly placed under *Instructions*.

- ReportBlock
- ForEach
- ForFiles

1.1.1. Attribues

The following table lists all attributes that can be used with the *Include* command.

Attribute Name	Description	required/optional
if	The specified file will only be included if the condition in this attribute is **true**. Here **true** means actually any of the following string values: • true • yes • on • 1	optional
ifNot	The specified file will only be included if the condition in this attribute is **NOT** true. For a description of the syntax of conditions see the *if* attribute.	optional
beforeVersion	The inclusion of the specified file will only be executed if the condition in this attribute is true. The condition specifies the name of a variable and a version number. The value of the variable then will be compared against the version number. The condition evaluates to true if the version	optional

	value of the variable is a lower version than the specified value. The variable name and the version to compare with must be separated by a colon (':'). Example: **beforeVersion="build.version:4** The following values for variable *build.version* will evaluate the example to true: • 1.0.45 • 4.3.27 • 4.2 The following values for variable *build.version* will evaluate the example to false: • 5.1.3 • 4.3.28 • 4.10.2 • 12.2.7 • 4.3.102	
sinceVersion	The inclusion of the specified file will only be executed if the condition in this attribute is true. The condition specifies the name of a variable and a version number. The value of the variable then will be compared against the version number. The condition evaluates to true if the version value of the variable is a higher or equal version compared to the specified value. The variable name and the version to compare with must be separated by a colon (':'). Example: **sinceVersion="build.version:3.** The following values for variable *build.version* will evaluate the example to true: • 3.0 • 3.21.7 • 3.21.11	optional

	• 11.0.1 • 3.123.2 The following values for variable *build.version* will evaluate the example to false: • 2.4.5 • 3.21.6 • 3.20.46.5	

1.1.2. Examples

Example	Description
<Include>/base/instructions/check12.cci</Include>	Inserts all commands from the instructions file named */base/instructions/check12.cci*
<Include if="servername='mx500'">*{INSTRUCTION_DIR}*/r	If the variable *servername* contains the value *mx500* then (and only then) the file *mx500_checks.cci* from the base directory for instruction files will be included.
<Include sinceVersion="build-version:3.5.1" beforeVersion="build-version:5.2.0">*{INSTRUCTI*	If the variable *build.version* a version that is equal or higher than *3.5.1* and lower than *5.2.0* then (and only then) the file *core_checks.cci* from the base directory for instruction files will be included.

1.2. ForEach

The purpose of this command is to repeat all enclosed instructions for all values specified in this command.

1.2.1. Attribues

The following table lists all attributes that can be used with this command.

Attribute Name	Description	required/optional
name	Specifies the name of the variable that contains the current value of the value list for each iteration. If this attribute is not defined the default variable name **_EACH_** will be used. See the Variables page to find	optional

	out what characters are allowed in variable names.	
values	Contains the list of values for which to loop over the enclosed instructions. For each value in this list the enclosed instructions will be executed once.	**required**
separator	Defines the separator in the list of values. If this attribute is not specified the default separator (i.e. comma ',') is used.	optional

1.2.2. Examples

```
<ForEach name="filename" values="test.jar:sample.jar:util.jar"
separator=":">
  <FileSystem>
      <AssertExistence label="{filename}"
element="/testdata/lib/{filename}"/>
  </FileSystem>
</ForEach>
```

Description: For each JAR file listed in the *values* attribute it will be checked if it exists in the directory /testdata/lib.

1.3. ForFiles

The purpose of this command is to iterate over a set of files or directories and execute all enclosed instructions for each file/directory found.

This command always sets some variables for each iteration (i.e. for each found file).

ABSFILENAME
Contains the absolute filename of the current iteration's file.
ABSDIRNAME
Contains the absolute directory name (i.e. without the file's name) of the current iteration's file.
RELFILENAME
Contains the relative file path of the found file. That is relative to the defined start directory in attribute *dir*.
RELDIRNAME
Contains the relative directory path of the found file. That is relative to the defined

start directory in attribute *dir*.
**FILENAME**
Contains the filename without any path information.
**DIRNAME**
Contains the relative directory path of the found file. That is relative to the current work directory.

1.3.1. Attribues

The following table lists all attributes that can be used with this command.

Attribute Name	Description	required/optional
dir	Specifies the directory from where to start searching for files matching the given pattern. If not set the search starts in the current working directory.	optional
pattern	Contains one or more name patterns that specify the filter for the files iterate over. A pattern can contain '*' for any number of any character and '?' for single occurance of any character. If more than one pattern is needed they must be separated by ';'.	**required**
recursive	Defines whether or not the file search should go recursivly throgh all sub directories. If not set the default value is "false".	optional
digit	Allows to specify a single wildcard character that represents a digit (i.e. 0-9). Usually '#' is used for this purpose, but any other character can be used as well.	optional
dirsOnly	If this attribute is set to "true" (or "yes, "on", "1") then the command iterates over directories rather than files.	optional

	If not set the default value is "false".	

1.3.2. Examples

```
<ForFiles dir="configuration/local" pattern="*.properties"
recursive="true">
  <FileSystem>
    <ReportValue label="Timestamp of {_FILENAME_}"
element="{_ABSFILENAME_}|@lastModified"/>
  </FileSystem>
</ForFiles>
```

Description: For each found properties file the corresponding timestamp of its last modification gets reported.

```
<ForFiles pattern="*.html" dir="stats" recursive="yes">
  ...
</ForFiles>
```

Description:

```
  Assuming that the current directory is c:/temp and the file
  c:/temp/stats/january/access.html was found with the above command
  then the variables will have the folowing values:

  _ABSFILENAME_    c:/temp/stats/january/access.html
  _ABSDIRNAME_     c:/temp/stats/january
  _RELFILENAME_    january/access.html
  _RELDIRNAME_     january
  _FILENAME_       access.html
  _DIRNAME_        stats/january
```

1.4. Set

The purpose of this command is to set the value of a variable. It can be used everywhere in an instruction file.
Each setting of a variable overwrites the previous value.
Since variables can have a global and a local scope it is possible to specify the scope with this command. However, it is only reasonable to use this feature when setting a global variable's value from inside a local context (i.e. from inside a data source adapter).

1.4.1. Attribues

The following table lists all attributes that can be used with this command.

Attribute Name	Description	required/optional
name	Specifies the name of the variable to be set. See the Variables page to find out what characters are allowed in variable names.	**required**
value	Any text of any size. Can even be an empty value.	**required**
scope	The scope can either be "global" or "local". If omitted it will be determined from the current context. That is, if the command is executed in a local context (i.e. inside a data source adapter tag) then the scope by default is "local", otherwise "global".	optional
asDefault	Setting this attribute to "true" allows setting the vaule of the variable as default only. That is, the value will only be assigned if the variable is not yet initialized.	optional

1.4.2. Examples

```
<Set name="basePath" value="sample/data/config"/>
```

Description: Sets the variable *basePath* to the current value *sample/data/config*.

```
<Set name="server.hostname" value="target.example.com" scope="global"/>
```

Description: Sets the global variable *server.hostname* to the current value *target.example.com*.

1.5. SetFrom

The purpose of this command is to assign the value of a configuration element to a variable.

Note:
This command must be used only inside a *Data Source Adapter* tag.

By default the variable is treated as *local* if not explicitly specified differently.

1.5.1. Attribues

The following table lists all attributes that can be used with this command.

Attribute Name	Description	required/optional
name	Specifies the name of the variable to be set. See the Variables page to find out what characters are allowed in variable names.	**required**
element	The element from which to retrieve the value that should be assigned to the variable. The syntax of this attribute depends on the *Data Source Adapter* inside this command is used. Refer to the documentation of the appropriate *Data Source Adapter*.	**required**
scope	The scope can either be *global* or *local*. If omitted it will be *local*.	optional
range	If the specified element returns more than one value then the range attribute can be used to specify which value(s) assign to the variable. Valid definitions for range are *all* or *first* or *last*. If omitted it will be *all*.	optional
separator	If for multiple values the range was set to *all* then this attribute can be used to define the separator between the values. If omitted the separator will be ','.	optional
default	Allows to specify a default value which will be assigned to the variable if the speified element cannot be found.	optional

	If this attribute is omitted and the element cannot be found then an error will added to the result report.	

1.5.2. Examples

```
<SettingsFile type="properties" name="base.properties">
  <SetFrom name="color_config_file" element="color.definitions"
scope="global"/>
</SettingsFile>
<SettingsFile type="ini" name="{color_config_file}">
  <ReportValue element="[Dialogs]/background"/>
</SettingsFile>
```

Description: Reads the filename of the ini-file that contains color configurations from property *color.definitions* in file base.properties and assigns it to variable *color_config_file*. Then it reports the current value of the *[Dialogs]/background* setting in this ini-file.

1.6. SetFromFile

With this command it is possible to load a set of variables from a properties file.
By default the loaded variables are added to the variable scope the command is executed in. Only if the command is running inside a *Data Source Adapter* (i.e. in local scope) it is possible to specify to load the variables to the global scope anyway.

1.6.1. Attribues

The following table lists all attributes that can be used with this command.

Attribute Name	Description	required/optional
file	Specifies the name of the properties file from which to read the variables.	**required**
scope	The scope can either be *global* or *local*. If omitted it will be the current execution scope.	optional

1.6.2. Examples

```
<SetFromFile file="set2.properties" scope="global"/>
```

Description: Reads all properties into the global variable pool. If some of the variables already exist their value will be modified.