Assertion Commands

Table of contents

1. Common Attributes

The following table lists all attributes the assertion commands have in common. However, keep in mind that particularly the syntax of the **element** attribute depends on the *Data Source Adapter* it is used with.

Apart from these common attributes each assertion command might have additional attributes that are specific for a particular *Data Source Adapter*. If such attributes exist they are documented with the corresponding *Data Source Adapter*.

Attribute Name	Description	required/optional
id	A unique identifier which will also be written to the report. It can be set to any value. Its main purpose is to associate a message in the report to its originating instruction in the instruction set. That allows easy tracking which reported value belongs to which instruction command.	optional
label	Sometimes the element's identifier is not quite human readable. In such a case it could be useful to define the *label* attribute with a name that a human being can understand. The contents of the *label* attribute is always copied unchanged to the report.	optional
element	The identifier for the element to check or report. **Note:** The syntax of this attribute depends on the *Data Source Adapter* it is used with. For an XML based *Data Source Adapter* that could be an XPath expression, for a Java properties file just a simple property name.	**required**
not	If this attribute is set to "true"	optional

	(or "yes, "on", "1") then the result gets negated. That's a simple way to express that something is expected *not* to be as specified.	
optional	If this attribute is set to "true" (or "yes, "on", "1") then the check is optional. That is, in case that the element cannot be found at all then this is not reported as an error or failure. However, if the element was found the check gets executed as usual and reports a failure if it wasn't successful.	optional
useSlashes	If this attribute is set to "true" (or "yes, "on", "1") then all backslashes in the found element's data will be translated into forward slashes. The default is "no".	optional
resultVar	If this attribute is set then the value of the assertion ("true" or "false") will be set to a variable with the name specified in this attribute. Whether the variable is "local" or "global" depends on the *scope* attribute.	optional
scope	Defines the scope of the variable name defined by attribute *resultVar*. The scope can either be "global" or "local". The default is "local".	optional
skipReporting	If this attribute is set to **"all"** or **"failure"** or **"error"** then the outcome of the assertion might not be added to the report. **all** Neither error nor failure will be reported **failure** The failure of the assertion	optional

	will not be reported **error** An error during the assertion will not be reported This option particularly makes sense in conjunction with the attribute *resultVar* so that the asstion result will be stored into a variable rather than causing a report entry. The default is "false".	

2. Common Child Elements for Multiple Values

The purpose of the following elements is to specify multiple values to check the found value(s) against.
There are several assertion commands that do not just check against a single pre-defined value but against a whole set of values.

Both elements can occur multiple times in any order. All their values are collected into a set which the found element(s) will be checked against.

Element Name	Description
Value	Specifies one value to check against.
ValueList	Specifies a list of values where the comma character is used as separator. However, the separator can be changed by setting the attribute *separator* to the character that should be used to delimit the single values in the list.

Example:

```
        <Assert..... element="...">
          <Value>200</Value>
          <Value>500</Value>
          <ValueList>330,340,350,380</ValueList>
          <Value>700</Value>
          <ValueList
separator="|">1000|2000|9000|11000|{number_list}</ValueList>
        </Assert......>
```

3. Commands

Note:
The following five commands automatically do an integer comparison if both, the expected and the actual value can be converted to integers. Otherwise, a string comparison is done.

- AssertEquals
- AssertGreater
- AssertGreaterOrEqual
- AssertLess
- AssertLessOrEqual

3.1. AssertEquals

The purpose of this command is to check whether or not the value of the specified element is equal to the value specified in the body this command. If the element's value is not equal to the specified value then a failure message (**msgid="FAIL0001"**) is written to the report.

3.1.1. Attributes

This command supports all the common attributes.

3.1.2. Example

```
<AssertEquals id="CHECK78" element="MaxConnections">25</AssertEquals>
```

Description: If the element *MaxConnections* ist not *25* then an assertion failure message will be written to the report.

```
<AssertEquals label="Server Name"
element="//Config/server[@id='M19']/host/@fqdn">
  www.testserver.com
</AssertEquals>
```

Description: If the element (i.e the server name) that is specified by the XPath expression is not *www.testserver.com* then an assertion failure message will be written to the report.

3.2. AssertLess

The purpose of this command is to check whether or not the value of the specified element is less than the value specified in the body of this command. If the element's value is not less

than the specified value then a failure message (**msgid="FAIL0002"**) is written to the report.

3.2.1. Attributes

This command supports all the common attributes.

3.2.2. Example

```
<AssertLess id="L210" label="Limit LDAP search result"
element="searchLimit">1000</AssertLess>
```

Description: If the element *SearchLimit* ist not less than *1000* then an assertion failure message will be written to the report.

3.3. AssertGreater

The purpose of this command is to check whether or not the value of the specified element is greater than the value specified in the body of this command. If the element's value is not greater than the specified value then a failure message (**msgid="FAIL0003"**) is written to the report.

3.3.1. Attributes

This command supports all the common attributes.

3.3.2. Example

```
<AssertGreater id="GR09" element="port">9000</AssertGreater>
```

Description: If the element *port* ist not greater than *9000* then an assertion failure message will be written to the report.

```
<AssertGreater element="Specification-Version">F<AssertGreater>
```

Description: If the element *Specification-Version* is not greater than *F* then an assertion failure message will be written to the report.

3.4. AssertLessOrEqual

The purpose of this command is to check whether or not the value of the specified element is

less or equal compared to the value specified in the body of this command. If the element's value is not less than or equal to the specified value then a failure message (**msgid="FAIL004"**) is written to the report.

3.4.1. Attributes

This command supports all the common attributes.

3.4.2. Example

```
<AssertLessOrEqual element="logLevel">3</AssertLessOrEqual>
```

Description: If the element *logLevel* ist not *3* or less then an assertion failure message will be written to the report.

```
<AssertLessOrEqual label="Max memory usage"
element="memory">1024<AssertLessOrEqual>
```

Description: If the element *memory* is not greater than *1024* then an assertion failure message will be written to the report.

3.5. AssertGreaterOrEqual

The purpose of this command is to check whether or not the value of the specified element is greater or equal compared to the value specified in the body of this command. If the element's value is not greater than or equal to the specified value then a failure message (**msgid="FAIL005"**) is written to the report.

3.5.1. Attributes

This command supports all the common attributes.

3.5.2. Example

```
<AssertGreaterOrEqual element="open.files">8</AssertGreaterOrEqual>
```

Description: If the element *open.files* ist not *8* or greater then an assertion failure message will be written to the report.

```
<AssertGreaterOrEqual element="//product[@name='tools']/@patch-level">
  4
```

```
<AssertGreaterOrEqual>
```

Description: If the element *//product[@name='tools']/@patch-level* is not greater than *4* then an assertion failure message will be written to the report.

3.6. AssertExistence

The purpose of this command is to check whether or not the specified element exists. If the element does not exist then a failure message (**msgid="FAIL0006"**) is written to the report.

3.6.1. Attributes

This command supports all the common attributes and additionally the following:

Attribute Name	Description	required/optional
emptyExists	If this attribute is set to "yes", an existing setting with an empty value is accepted as existing. If set to "no", such an empty value will cause a configuration failure message. If the attribute is not explicitly set the default value is "yes".	optional

3.6.2. Example

```
<AssertExistence label="proxy module"
element="LoadModule[@1='proxy_module']/@2"/>
```

Description: Asserts that the LoadModule proxy_module with a second parameter exists. If not a failure message is added to the report.

3.7. AssertMatch

With this command a value can be checked against a simple pattern. The pattern can be any string. In such a pattern the characters '*', '?' and '#' have specials meanings. The '*' stands for any number and any character. The '?' represents any single occurance of an arbitrary character. The '#' represents a single digit (i.e. 0-9). If the element does not match the pattern then the failure message (**msgid="FAIL0008"**) is added to the report.

3.7.1. Attributes

This command supports all the common attributes.

3.7.2. Example

```
<AssertMatch id="D420" label="Mail address"
    element="[cn=jdoe,ou=users,dc=company]/@mail">*.*@*.*</AssertMatch>
```

Description: If the LDAP element with the distinguished name *cn=jdoe,ou=users,dc=company* must have an attribute named *mail* with a typical eMail address value that is matching the pattern *.*@*.*. The value *john.doe@company.com* will be fine but the value *jdoe@company.com* will cause an assertion failure message being added to the report.

3.8. AssertContains

This command allows to treat the value of elements as a list of values. It can assert that a particular value is in such a list. The default separator for the list elements is comma (','). If the list value does not contain the expected value then the failure message (**msgid="FAIL0009"**) is added to the report.

3.8.1. Attributes

This command supports all the common attributes.

3.8.2. Example

```
<AssertContains id="ABC" label="Italian"
    element="supported.languages" case-sensitive="no">it</AssertContains>
```

Description: If the property *supported.languages* doesn't contain the value "it" (case insensitive comparison) then a failure message gets reported.

3.9. AssertOneOf

This command allows to check if the value of an element is equal to at least one of a list of allowed values. If the element's value is not equal to any of the allowed values then failure message (**msgid="FAIL0010"**) is added to the report.
The values to check against have to be specified by child elements <Value> and <ValueList>.
Both child elements can be used as often as necessary and in arbitrary order. Find the detailed description here.

3.9.1. Attributes

This command supports all the common attributes.

3.9.2. Examples

```
<AssertOneOf label="Language" element="current.language"
case-sensitive="no">
  <Value>it</Value>
  <Value>fr</Value>
  <Value>de</Value>
</AssertOneOf>


<AssertOneOf label="Language" element="current.language"
case-sensitive="no">
  <ValueList>it,fr,de</ValueList>
</AssertOneOf>
```

Description: Both examples are equivalent. They are checking if the property *current.language* is "it" or "fr" or "de" (case insensitive comparison). If that is not the case then a failure message is added to the report.

3.10. AssertMatchOneOf

This command is useful to check if the value of an element matches at least one of a list of specified patterns. If the element's value does not match any of the defined patterns then failure message (**msgid="FAIL0018"**) is added to the report.
Currently only **'*'** and **'?'** are supported in patterns. Regular expressions are not yet supported.
The values to check against have to be specified by child elements <Value> and <ValueList>.
Both child elements can be used as often as necessary and in arbitrary order. Find the detailed description here.

3.10.1. Attributes

This command supports all the common attributes.

3.10.2. Examples

```
<AssertMatchOneOf label="Port" element="redirect.url" case-sensitive="no">
  <Value>http://*:8080/*</Value>
```

```
  <Value>https://*:9443/*</Value>
  <Value>http://*:81/*</Value>
</AssertMatchOneOf>


<AssertMatchOneOf label="Port" element="redirect.url" case-sensitive="no">
  <ValueList
separator=";">http://*:8080/*;https://*:9443/*;http://*:81/*</ValueList>
</AssertMatchOneOf>
```

Description: Both examples are equivalent. They are checking if the property *redirect.url* contains a URL with one of the port numbers "81" or "8080" or "9443". If that is not the case then a failure message is added to the report.

3.11. AssertMatchAll

With this command it is possible to ensure that the value of an element matches **all** of a list of specified patterns. If the element's value does not match any one of the defined patterns then failure message (**msgid="FAIL0016"**) will be added to the report.
If the assertion is negated (not="true") then failure message (**msgid="FAIL0015"**) will be added to the report if the found element's value matches at least one of the specified patterns.
Currently the wildcard characters **'*'** and **'?'** are supported in patterns. Regular expressions are not yet supported.
The patterns to check against have to be specified by child elements <Value> and <ValueList>.
Both child elements can be used as often as necessary and in arbitrary order. Find the detailed description here.

3.11.1. Attributes

This command supports all the common attributes.

3.11.2. Examples

```
<AssertMatchAll not="true" label="Logo" element="image.main.logo"
case-sensitive="no">
  <Value>*.gif</Value>
  <Value>*.jpg</Value>
  <Value>*.png</Value>
</AssertMatchAll>


<AssertMatchAll not="yes" label="Logo" element="image.main.logo"
case-sensitive="no">
  <ValueList>*.gif,*.jpg,*.png</ValueList>
</AssertMatchAll>
```

Description: Both examples are equivalent. They are checking if the property *image* matches none of the file name patterns "*.gif" or "*.jpg" or "*.png". If the property matches one of the patterns then a failure message is added to the report.

3.12. AssertNewLine

This assertion command is only reasonable within a text file data source adapter. It can be used to check the end-of-line characters in a text file. If at least one line is terminated by a different one than the expected the failure message (**msgid="FAIL0011"**) is added to the report.

3.12.1. Attributes

Attribute Name	Description	required/optional
element	The element name for this assertion command must always be **EOL** (i.e. "End Of Line").	**required**
value	The value attribute specifies the expected line-end character(s). Valid values are: • "CR" -> carriage return • "CRLF" -> carriage return/ line feed • "LF" -> line feed • "LFCR" -> line feed/carriage return	**required**

3.12.2. Example

```
<AssertNewLine label="Unix style line end" element="EOL" value="LF"/>
```

Description: Checks all lines in a file to end with line-feed (LF) character. If any line has different line-end character(s) then the assertion fails.