Groovy Script

While automating web services in SoapUI you may come across different situations where it is required to write groovy scripts to perform different actions.

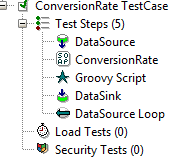
We assume that you are aware of data driven testing in SoapUI.

# Response Validation

## Case #01: To compare the response from service with the expected value.

For example, your service returns the conversion rate from one currency to another currency (provided you have access to all the conversion rates in the database). You have to validate whether the values returned by service are same as the conversion rate in the database. So you will be having predefined values for conversion rate (In DataSource) and will compare this value with the actual response from service.

**Set-Up:**



**Script (Groovy Script):**

//declare Groovy Utils context

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

//declare variable for Expected Rate

def expectedRate = context.expand( '${DataSource#ExpectedRate}' )

//declare response xml holder (holds complete response xml)

def holder = groovyUtils.getXmlHolder("ConversionRate#Response")

//get value from response xml using XPATH

def actualRate = holder.getNodeValues("//\*:ConversionRateResult")

//to write data into data sink

def runnerComplete = testRunner.testCase.testSteps.DataSink

if (expectedRate.equals(actualRate)){

TestStatus = "Pass"

} else {

TestStatus = "Fail"

}

//write result to dataSink

runnerComplete.setPropertyValue("TestStatus",TestStatus)

## Case #02: To trim time stamp from response date value.

If the response date value contain date with time stamp, let’s say 2015-08-01+05:00 while comparing this date with expected data the test may pass in one environment and the same test may fail in different environment located in different time zone (2015-08-01+01:00)

**Script (Groovy Script):**

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def holder = groovyUtils.getXmlHolder("ConversionRate#Response")

//trim time stamp from response value using method trimTimeStamp

def conversionDate = CommonUtils.trimTimeStamp(holder.getNodeValues("//\*:ConversionDate"))

//set updated value into data sink

def runnerComplete = testRunner.testCase.testSteps.DataSink

runnerComplete.setPropertyValue("ConversionDate", conversionDate)

NOTE: The method CommonUtils.trimTimeStamp is called from SoapUI external script, the details of this method is available in the below path (here CommonUtils is the class name which contains method trimTimeStamp())

* <https://fswiki.evry.com/display/projects/Custom+Groovy+Library+files+at+common+repository>

[Click here (Ctrl + click) for details on setting external script library](#_Add_Script_library)

# Request Manipulation:

## Case #01: To set input date always as today’s date.

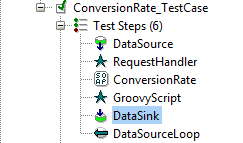
If you have an input date field which accept current date as valid input and in dataSource you have the date value as ‘2015-07-21’. The test will run successfully for this particular date and the next day same test will through exception since this date will be past date.

## Case #02: To set input date as past/future date.

If you need to set an input date as past date (directly we can set it as past date). If you want to set the future value, let’s say you set date value in dataSource as ‘2016-07-21’. The test which run successfully until this particular date is reached, after this date the same test will fail.

You can include a groovy script (RequestHandler) to convert input date value and set it to test request.

**Set-Up:**



**Script (RequestHandler):**

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def holder = groovyUtils.getXmlHolder("ConversionRate#Request")

def runnerComplete = testRunner.testCase.testSteps.DataSink

//define data from dataSource

def conversionDate = context.expand( '${DataSource#ConversionDate}' )

//convert date if value is not blank and if length is less than 4 digits

if(!conversionDate.equals("") && conversionDate.size() <= 4){

conversionDate = CommonUtils.getDate(conversionDate)

}

//set date value to request and dataSink

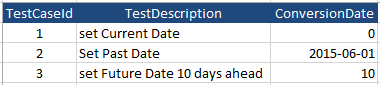
holder.setNodeValue("//\*:conversionDate",conversionDate)

runnerComplete.setPropertyValue("ConversionDate",conversionDate)

NOTE: The method CommonUtils.getDate is called from soapui external script, the details of this method is available at <https://fswiki.evry.com/display/projects/Custom+Groovy+Library+files+at+common+repository>

EX:

If the dateValue in input excel sheet are as below:



Output from script to test request will be:

1. Current date (converts 0 to today’s date i.e. today+0 days)
2. Set past date value same as input value (length is greater than 4 char)
   1. Also you can use negative numbers to get past date (ex: -10 will give date 10 days back)
3. Converts 10 to date value 10 days ahead of current date.

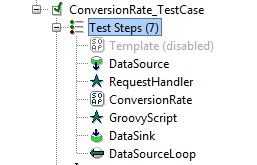
## Case #03:Set request xml with Template xml and update request as per input values

If you have an input xml field with option Add/Delete option as shown below. You need to delete this if input values for this input are blank in input file (DataSource). Note that once the node is deleted from request XML you cannot retrieve it back (in automation execution), so we need to copy complete request XML file from a template and set this to request xml.



**Set Up:**

* Add new test request to test suite and name it as Template (convention).



**Script (RequestHandler):**

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def holder = groovyUtils.getXmlHolder("ConversionRate#Request")

//define template request

def requestTemplate = groovyUtils.getXmlHolder("Template#Request")

//set template request to test request (holder)

holder.getXmlObject().set(requestTemplate.getXmlObject())

holder.updateProperty()

//remove node foreignCurrencyCreate if input currencyCode is blank

def currencyCode = context.expand( '${DataSource#CurrencyCode}' )

if(currencyCode.equals("")){

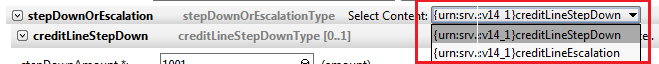
holder.removeDomNodes('//\*:foreignCurrencyCreate')

}

holder.updateProperty()

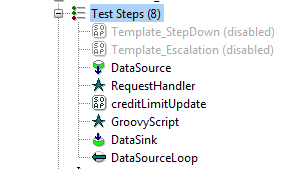
## Case #04: Set request xml if input contain choice of two or more objects.

If the input request contain choice of two or more objects as shown below. You need to write script that can select one of this input depending up on the input data.



**Set Up:**

* Add separate template for creditLineStepDown and creditLineEscalation details. As shown below. Depending up on the input data set different template to request.



Consider that if the input data contains data for stepdown, it should have input data for stepdownAmount node (request node)

**Script (RequestHandler):**

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def holder = groovyUtils.getXmlHolder("creditLimitUpdate#Request")

//define template request

def template\_StepDown = groovyUtils.getXmlHolder("Template\_StepDown#Request")

def template\_Escalation = groovyUtils.getXmlHolder("Template\_Escalation#Request")

//set different templates depending upon input data

def stepdownAmount = context.expand( '${DataSource#StepdownAmount}' )

// if stepdownAmount is not blank set StepDown template

if(!stepdownAmount.equals("")){

holder.getXmlObject().set(template\_StepDown.getXmlObject())

} else{

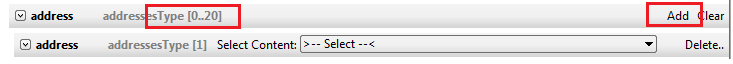
holder.getXmlObject().set(template\_Escalation.getXmlObject())

}

holder.updateProperty()

## Case #05: Add multiple xml objects to request

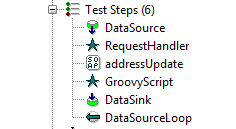
If you have input where the user can add multiple input object as shown below.



Here the user can add up to 20 address details in single request. To perform this in automation it is required to write some script.

NOTE: Input data for this scenario will be fed as comma separated values from dataSource (ex: 0123,0234 )

**Set Up:**



**Script (RequestHandler):**

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def holder = groovyUtils.getXmlHolder("addressUpdate#Request")

//define address data from dataSource

//input data will be comma seperated vaules ex: areaCode as 0123,0234

def streetName = context.expand( '${DataSource#StreetName}' ).split(",")

def cityName = context.expand( '${DataSource#CityName}' ).split(",")

def areaCode = context.expand( '${DataSource#AreaCode}' ).split(",")

//get xml object node – parent node containing all child nodes

def node = holder.getDomNode("//\*:address")

def length = streetName.size()

//update request according to input data

// add new node and set value

for( int i=0; i<length; i++)

{

if(i > 0){

def newNode = node.cloneNode(true)

node.getParentNode().insertBefore(newNode, node.getNextSibling())

}

holder.setNodeValue("//\*:streetName", streetName[i])

holder.setNodeValue("//\*:cityName", cityName[i])

holder.setNodeValue("//\*:areaCode", areaCode[i])

}

holder.updateProperty()

EX 2:

//update request according to input data

for( int i=0; i<lengthOfPriceMatrixList; i++) {

log.info("more than one object")

if(i > 0){

def newNode = nodePriceMatrix.cloneNode(true)

nodePriceMatrix.getParentNode().insertBefore(newNode, nodePriceMatrix.getNextSibling())

}

holder.setNodeValue("//\*:parameterId",parameterId[i])

holder.setNodeValue("//\*:parameterSeqNumber",parameterSeqNumber[i])

holder.setNodeValue("//\*:parameterClass",(parameterClass.size() > i) ? parameterClass[i] : "")

holder.setNodeValue("//\*:parameterClassOriginal",(parameterClassOriginal.size() > i) ? parameterClassOriginal[i]: "")

holder.setNodeValue("//\*:parameterValue",(parameterValue.size() > i) ? parameterValue[i]: "")

}

holder.updateProperty()

# Add external script library:

Need for Script Library: Used to save common methods in an external class file and can be referred in different groovy script test step. This will increase reusability.

**Steps:**

* Make new folder (e.g. "C:\GroovyLib"), Add groovy class file into the folder, “CommonUtils.groovy”.



* Set File > Preferences > SoapUI Pro tab > Script Library to library folder. (I would set that to "C:\GroovyLib" in my example.)
* Then restart SoapUI Pro to pick up the library script.
* Call groovy script methods as follow(in groovy script test step)

**Script:**

//call CommonUtils.groovy methods

log.info CommonUtils.getDate(“10”)

# Add xml node into soap request:

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def holder = groovyUtils.getXmlHolder("customerRelationCreate#Request")

def parentNode = holder.getDomNode("//\*:qualification")

def childNode = holder.getDomNode("//\*:targetCustomerKey")

def requestDoc = parentNode.getOwnerDocument()

def namespace = childNode.getNamespaceURI()

def specificiteNode = requestDoc.createElementNS(namespace, 'sourceCustomerKey')

def specCoefComptageNode = requestDoc.createElementNS(namespace, 'customerNumber')

specificiteNode.appendChild(specCoefComptageNode)

//def nodeText = '''<sourceCustomerKey>

// |<customerNumber>?</customerNumber>

// |</sourceCustomerKey>'''.stripMargin()

//def nodeHolder = new com.eviware.soapui.support.XmlHolder( nodeText )

//def nodeItem = nodeHolder.getDomNode( "//sourceCustomerKey" )

def importNode = requestDoc.importNode( specificiteNode, true )

parentNode.insertBefore( importNode, childNode )

holder.updateProperty()

# getProperty from different testCase in SoapUI test suite

def propertyValue = testRunner.testCase.testSuite.testCases["ConversionRateTestCase"].getTestStepByName("ConversionRate").testRequest.response.getResponseHeaders()["Content-Type"][0]