Documentation of

Automation for SDM Client and SMGR SDM

Prepared By – Bhumit Shingala

Under Supervision of Dhanashree Gaigaware

Table of Contents

Table of Contents

Preface: (Must Read)

- SDM Client / SMGR SDM is built on ext-js which leverages HTML5 features on modern browsers while maintaining compatibility and functionality for legacy browsers.
- All the ID and xPaths which are rendered on DOM in runtime are generated dynamically from ext-js in SDM Client/SMGR SDM, is the biggest hurdle for the automation suite.
- To mitigate the issue, methods are written in such a way that it will take ID and xPaths dynamically in runtime and then doing all the operations. For that, JS (JavaScript) is injected into the automation code.
- As resources (IP and host capacity) are less and it is requirement to perform all the tests (functional testing and unit testing) for all the elements (Avaya supported), code is written multi-threaded to run simultaneous threads (installation) and perform execution for all the use cases as TestNG tests.
- Individual operations can be performed as well like - only host related operations, only location related operations, installation of only one VM and operations related to it.

Pre-Requisites to use Automation:

Necessary Arrangements (Relocate and check all necessary files):

Please follow the steps given below –

Please check whether following files are available or not –

1. Object repository file in - . /Third party/OR/xprev.properties. (All static IDs and xPaths will be retrieved from here.)
2. The folder - . /Third Party/tempXMLs which must have atleast one file "temp.xml". (All the XML files for different elements will be generated from this file. (base format))
3. The folder - ./Third Party/Input Files – which must have atleast 6 files – **input.properties, inputsm.properties, black-whitelistedIP.properties, inputelem.txt, inputip.txt, Inputtemp.txt** and two non-mandatory files **– ovanames.txt,js.txt**
4. Concurrent.xml file in Project root folder.
5. Log4j.xml file in project root folder.
6. Make a firefox profile to run automation in that profile. Go to run and then type firefox.exe –p and hit enter. Pass this profile name for selecting profile in code.

For your Information:

Description related to different files:

- Input.properties – Input file related to all operations except VM related inputs.
- Inputsm.properties – Input file related to VM related inputs.
- Ovanames.txt – Contains names of all OVAs which are downloaded. (It will be generated in runtime, please refer to **maintainedList** method).
- Black-whitelistedIP.properties – Contains state of IP (blacklist or whitelist). If whitelist then it can be used to deploy OVA otherwise it cannot be used.
- inputelem.txt – Names of all elements.
- inputip.txt – Maximum available list of IPs.
- Inputtemp.txt – Counter, for which OVA should be deployed in next iteration.

Default Credentials for elements:

Utility Services (US) Username = admin

Utility Services (US) Password = admin01

AES Username = admin

AES Password = admin123

SAL Username = admin

SAL Password = admin01

How to run automation:

Pre-Checks:

Mandatory:

- Make sure that IPs you have given in inputip.txt must have an entry in black-whitelistedIP.properties and all are having values as “WhiteList”.
- Make sure that inputtemp.txt file should have value as ‘1’, because it will take the next element from inputelem.txt which is lined as the value given in inputtemp.txt.
  For e.g. If the value is 1 – it will read first line from inputelem.txt (technically second as line numbers start from 0) and it will install that element in next thread.

Non-Mandatory:

- Make sure that you don’t have any hosts added because if they are, maybe their names will be conflicted and tests related to hosts will fail.

- Just navigate to “driverConcurrentInstallations.java” and simply run that Java file.

Insights (Description) of Methods:

confirmDialogueBox (WebDriver driver):

This method confirms every dialogue box. (pass the driver instance)

In some dialogue box, we need to confirm twice. So I have added try-catch block for the second dialogue-box. If second dialogue box will appear then it will confirm it otherwise it will resume.

chooseLink (WebDriver driver, String Name, String vmOrHost, String linkText)

This method will choose the link like “Status Details”, “Installation Details” for given VM or Host. It will find that link in that particular row (which is passed as parameter ‘Name’ – VM name or Host Name).

Pass driver instance, name of VM or Host, ‘VM’ or ‘Host’ and link text like – “Status Details”.

maintainedList (WebDriver driver, String ID)

This method will write newest names of OVAs (newest version) in ovanames.txt so that we don’t have to hardcode version of any element and it will take that name from the file while installing.

Pass driver instance and ID of that combobox.

getViewFrame (WebDriver driver, String input)

This method will return Frame ID for given inputs. Check next method for more info.

findIDandFillValues (WebDriver driver,String filename,String input)

This method will automatically find all the fields which are to be filled and it will fill the values for the same for any given frame. If new feature comes in, we just have to add frame name in previous method and fill the input file. Method will automatically fill the values for the same. We don't have to write code and even we don't have to care for the ID. It will automatically be taken care of by the method.

For e.g. Location adding – just find the view frame and add it in getViewFrame. And then add inputs in input files like (Append 'AddLocation' or related arguments before the input) in input.properties file –

AddLocationAddress: = Tower-11,

AddLocationCity: = Pune,

AddLocationState/Province/Region: = Maharashtra,

AddLocationCountry: = India,

AddLocationZip/PostalCode: = 411028,

AddLocationName: = testLoc.

checkSuccessOrFailure (WebDriver driver, By locator, String vmName, int timeOutForException, boolean b, int noOfTimeOut, int counter, String linkText)

This method will check for Completion or failure for certain operations. As Selenium does not have any method to check for two types of text like – “Completed” and “failed” simultaneously in an element, this method is developed. It can check for either completion or failure one by one.

Pass the driver instance, locator for the link in which the text will be checked, VM name, timeout in seconds (how often we want to switch and check for success or failure – ideal is 5 to 7 seconds), Boolean true every time, no of timeouts (total timeout to perform operation within that timeout), counter always zero and linktext like – “Status Details”.