










Krypton Automation Product User Manual

Section Summary

 Introduction	 Quick Start
 Supported Platforms	 Execution Environment
 How to Install Krypton Setup	 How to write Test Cases
 Testing Process	 Parameters Files
 Keywords	

INTRODUCTION

Krypton is a generic framework which has been specifically designed to cater to needs of any client who seeks a low cost end-to-end test automation solution for web using open source tools. Krypton is designed in such a manner that the testers do not need to possess any scripting background at all to automate complex testing scenarios. Everything is already built-in and can be configured easily.

PRIMARY CHARACTERISTICS OF KRYPTON:

1. Should support multiple web browsers: Internet Explorer, Firefox, Safari, Chrome etc.
2. Should incorporate the 4Rs: Repeat-ability, reliability, re-use and robustness.
3. Should be a user-friendly solution

KRYPTON IS COMPOSED OF THE FOLLOWING COMPONENTS:

1. Test Engine: is the brain of the framework which will handle all other components
2. Test Driver: will drive automation tools like Selenium integrated with Krypton framework
3. Reporting Engine: will generate reports/log files for automation execution and email notifications.
4. Test Manager: will be responsible to retrieve data from different test management source systems (E.g. File System).

SUPPORTED PLATFORMS

1. Supported Operating Systems

- Windows (XP, Vista, Win 7 , Win 8.1 & Win10)

2. Supported Browsers

- Google Chrome 30 or later
chromedriver.exe is already included in setup. After installation, It will be in output folder.
 - Internet Explorer 10.0 or later
 - Mozilla Firefox 19.0 or later (upto 40.0.3)

HOW TO INSTALL KRYPTON SETUP

Enough of background study? Let's get started with the product:

Here you will be acquainted with how to quickly configure and run your test cases. Some sample test cases are provided along for

easier derstanding with installation package. Everything is pre-configured for the sample test cases.

LET'S START

Ensure all pre-requisites, download Krypton & run setup file and follow steps.



Click on 'Next'



Click on 'Next'



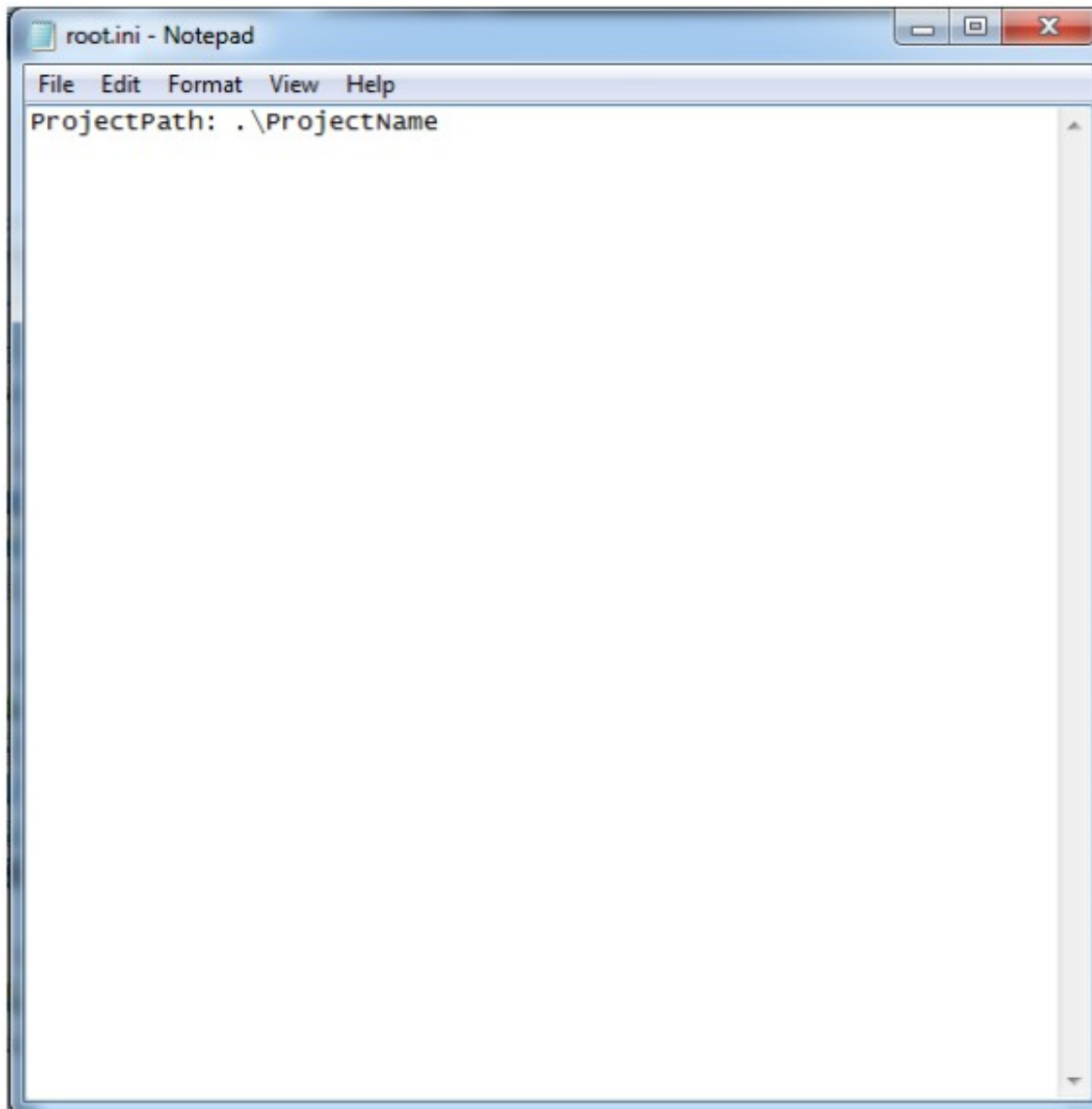
Wait while installation completes. Then the 'Confirm Installation' shows up.



Click on 'Next'



And the installation is complete.



By default Krypton will be installed under C drive.

Sample Test cases and its Object Repository is already provided under TestCases and Object_Repository folder respectively under Krypton folder.

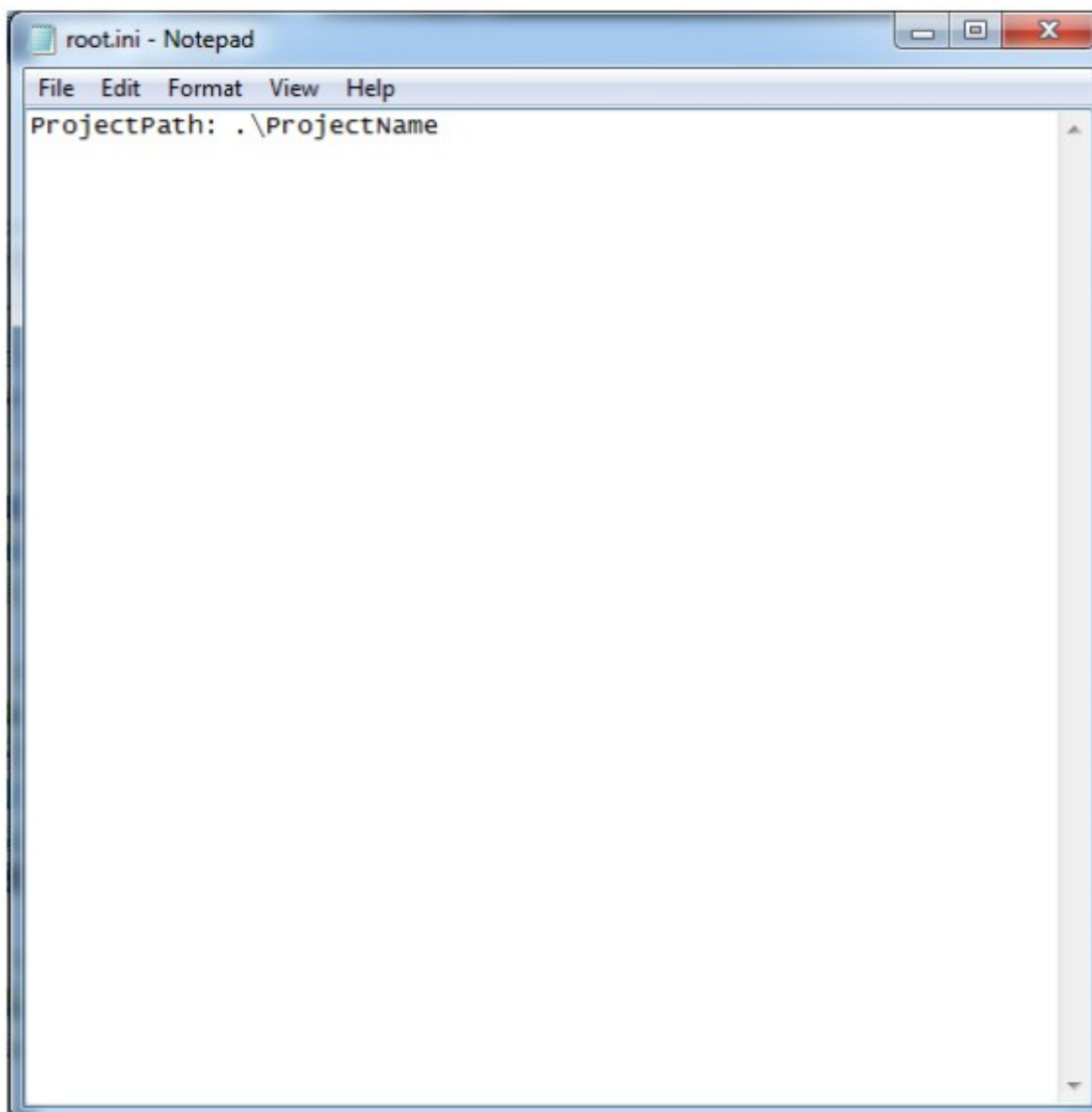
Run the executable file for execution of sample test cases

After execution results will be uploaded at Results folder under respective Project folder.

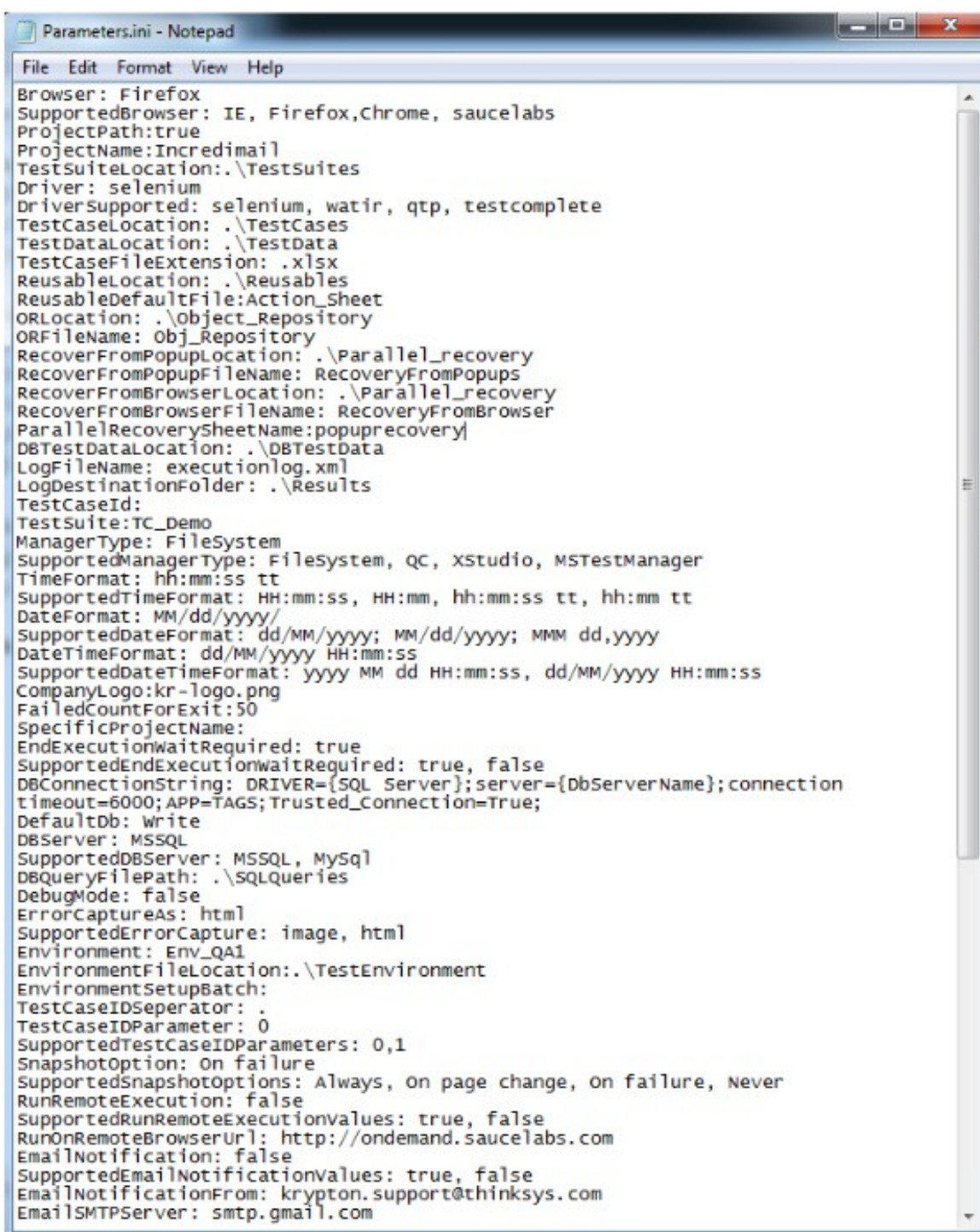
START WITH WRITING NEW TEST CASES AND CONFIGURE THE SETTING FOR EXECUTION

Create new Test case spreadsheet same format as of SampleTest case under TestCases folder. Write reusable test case in Action_sheet.xlsx file if any in same format placed under TestCases folder.

Identify the objects and create new objects in Obj_Repository.xlsx file in same format placed at Object_Repository folder.



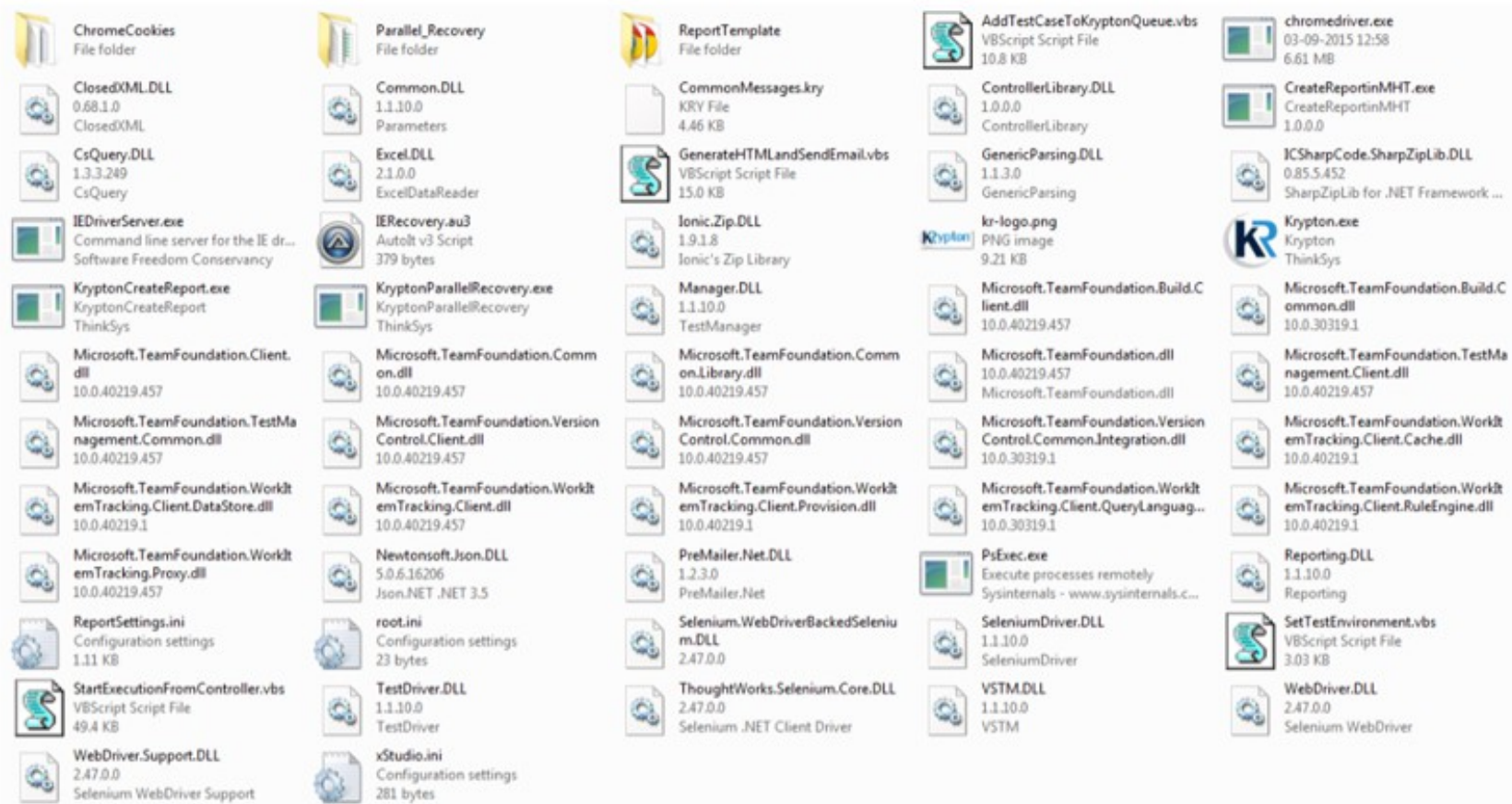
Mention test case id and Configure parameters.ini file accordingly and run Krypton.exe



The project folder gets created outside the Krypton folder with name specified in root.ini file.

Demo	02-09-2015 17:30	File folder	
Output	03-09-2015 18:06	File folder	
Krypton.exe	03-09-2015 17:27	Shortcut	3 KB
root.ini	03-09-2015 17:27	Shortcut	3 KB

The Krypton folder and its inside looks like this.



TESTING PROCESS

Testing with KRYPTON involves 4 main stages:

Creating Tests

1. Now write your test cases in a Excel sheet.
2. Prepare Object repository in a Excel sheet (Obj_Repository.xlsx).

Configuring Test:

After writing your test cases and preparing object repository you need to configure your test cases. For configuration follow the steps given:

1. Go to the installation folder of KRYPTON and search for the "parameters.ini" file.
2. Open the file and change following parameters according to your Test:
 - o LogDestinationFolder
 - o TestCaseId
 - o TestSuite

Running Test:

After configuring you can run your test by just running KRYPTON.exe file or double clicking KRYPTON shortcut on your desktop. You can also set parameters by sending them through command line. E.g. C:>KRYPTON.exe testcaseid=abc.tc01

Analyzing Test Results:

After test execution completed reports are generated in the **Results** folder. Reports are generated in two forms

1. **Zip form:** All HTML reports are zipped in a file. User can use these reports by unzipping the zipped file.
2. **Unzip form:** All HTML reports are placed in Results folder by the KRYPTON Reporting Engine

KEYWORDS

Krypton Keyword are words used in TestCases for specific operations. All Keywords used by Krypton are given below.

S.NO	KEYWORD	DESCRIPTION	PARENT	TEST OBJECT	DATA	
BROWSER OPERATIONS						
1.	openbrowser	Opens the browser. Without using 'Test Data', it redirects to a blank browser. However, with 'Test Data', it redirects to the URL specified in 'Data Column'			Test Data :- URL (optional)	
2.	clearbrowsercache	Clear Browser cache				
3.	closebrowser	Closes running Browser.				
4.	closeallbrowsers	Closes all the running Browsers.				
5.	refreshbrowser	Refresh the Browser.				
Navigational Keywords						
1.	navigateurl	Navigate to a particular URL specified in 'Data Column'.			URL as Data	
2.	goforward	Navigate to the next Page.				
3.	goback	Navigate to the previous Page.				
Web Form Operations						
1.	clear	Clear the text already present in a text Box.	Need Parent	Need Object	Test	
2.	enterdata	Enters Text in the Text Box. The text to be entered is given in the 'Data' Column	Need Parent	Need Object	Test	Text to be entered
3.	enteruniquedata	Generates and enters Unique Data in the given Object. The length of the Unique Data to be generated is given in 'Data' Column	Need Parent	Need Object	Test	Length of Data to be generate
4.	click	Click on a specific Object.	Need Parent	Need Object	Test	
5.	check	Check the CheckBox.	Need Parent	Need Object	Test	
6.	uncheck	Uncheck the CheckBox.	Need Parent	Need Object	Test	
7.	selectitem	Select a specific option from a List. The Object is the locator of the list. The Option to be selected is given in 'Data Column'.	Need Parent	Need Object	Test	Text of the item to be select in dropdown
8.	selectmultipleitems	Select multiple item in multiple selection box	Need Parent	Need Object	Test	Text of the items to be select separated by comma
9.	selectitembyindex	Select a specific option from a List. The Object is the locator of the list.	Need Parent	Need Object	Test	Index number of Item to be

		The Index of the Option to be selected is given in 'Data Column'.				select
Wait Keywords						
1.	wait	Pause's the execution for a static time which is mentioned in 'Data Column'.				Time should be in seconds
2.	waitforobject	Waits for particular Object to Appear.	Need Parent	Need Object	Test	Time out in second(optional)
3.	waitforobjectnotpresent	Waits for a particular Object to Disappear.	Need Parent	Need Object	Test	Time out in second(optional)
4.	waitforproperty	Waits for Property to appear. The property to wait for and the timeout is given in the 'Data Column'. Unit of timeout is seconds. Syntax — property timeout	Need Parent	Need Object	Test	property timeout in second
Alert Handling						
1.	acceptalert	Accepts the Alerts.(Applicable for browser Alerts)				
2.	dismissalert	Cancels the Alerts.(Applicable for browser Alert)				
3.	verifyalerttext	Verify text of the alert. The text to be verified is given in the 'Data Column'.				Text to be verify
Verify Keywords						
1.	verifytextcontained	Verify that specified text is present on current test object. The text to be verified is given in the 'Data Column'. If the object is not found in 'Object Repository', it verifies the text (given in data column) present on the entire page.	Need Parent	Need Object	Test	Text to be verify
2.	verifytextnotcontained	Verify that specified text is not present on current test object. The specified text is given in the 'Data Column'.	Need Parent	Need Object	Test	Text that verify as not present
3.	verifytextonpage	Verifies that the specified text is present on current web page. The text to be verified is given in the 'Data Column'.				Text to be verify
4.	verifytextnotonpage	Verify that specified text is not present on current web page. The text is given in the 'Data Column'.				Text that verify as not present
5.	verifylistitempresent	Verify that List option is present in focused List	Need Parent	Need Object	Test	Text of item present in dropdown
6.	verifylistitemnotpresent	Verify that List option is not present in specified List object	Need Parent	Need Object	Test	Text of item present in

						dropdown
7.	verifyobjectpresent	Verifies that the specified test object is present on current web page.	Need Parent	Need Object	Test	
8.	verifyobjectnotpresent	Verifies that the specified object is not present on current web page.	Need Parent	Need Object	Test	
9.	verifyobjectproperty	Verifies a specific property of the given Object and matches it with specified value. Syntax - PropertyName ValueOfProperty (To be given in 'Data Column').	Need Parent	Need Object	Test	PropertyName ValueOfProperty
10.	verifyobjectpropertynot	Verify the given property is not for Object	Need Parent	Need Object	Test	PropertyName Value Of Property that is not present
11.	verifypageproperty	Verify page url \ title				url or title
12.	verifytextinpagesource	Reads the html of the page, and searches for the specified 'text'. The text is given in 'Data Column'.				Text to be verify in page source
13.	verifytextnotinpagesource	Verify the Given Text is not present in page source. The text is given in 'Data Column'.				Text that verify as not present in page source
14.	verifypagedisplayed	Verify that current web page is display properly	Need Parent	Need Object	Test	
15.	verifyobjectdisplayed	Verifies if the object is being displayed or not on current web page				
Database Operations						
1.	executedatabasequery	Execute a database query against specified database server and environment. The query is given in 'Data Coloum' or in SQL folder.				Need sql Query
2.	getdatafromdatabase	Get all the record from database using specified query				Need sql Query
3.	getuserfortesting	Get all the record from database using specified query but if no record found then test will terminate instantly				Need sql Query
XML Operations						
1.	findxmlattribute	To find an attribute in XML				attribute name
2.	verifyxmlattribute	verify the Xml attribute is present				attribute name
3.	countxmlnodes	Used to Find the Count of Given Xml Node				XML node
4.	verifyxmlnodecount	Used to verify the Count of Given Xml Node				XML node count

Events and Actions

1.	mousemove	To focus and move the cursor to a specified Object.	Need Parent	Need Object	Test	
2.	mouseclick	To click on a Specific Object.	Need Parent	Need Object	Test	
3.	mouseover	To Hover the cursor on a Specific Object.	Need Parent	Need Object	Test	
4.	keypress	Used to press a Specific Key on a particular Object. The key to be pressed is given in 'Data Column'.	Need Parent	Need Object	Test	Key name: e.g . ENTER
5.	fireevent	Used to focus on an Object and perform an event mentioned in the 'Test Data'	Need Parent	Need Object	Test	Test Data:- (Event name e.g. blur)
6.	addaction	It saves an Action in an Array in the form of list. The action is given in the 'Data Column'.	Need Parent	Need Object	Test	e.g. click
7.	performaction	It executes the list of actions saved in the add action list.				

Miscellaneous

1.	runexceltestcase	Runs a particular test case. The test case ID to be run is given in the 'Data Column'				Test Data- (Test Case ID of the TestCase to run)
2.	settestmode	Similar to If Else statements. To set A/B Testing mode based on certain conditions.	Need Parent	Need Object	Test	<Mode Name> value
3.	getpageproperty	Stores the page Url/Title in a given variable. The variable is given in the 'Data Column'.				Variable to store and the url/Title (e.g url)
4.	executestatement	Execute a java script				Java Script
5.	setvariable	It sets a value to a variable				<Varibale name> value
6.	setparameter	Used to override the Krypton parameter at run time				<Krypton parameter> value
7.	requestwebservice	Execute an API Query with specified Method				Api query
8.	downloadfile	It downloads the file to a particular location in the System and save the location in a variable 'downloadedfile'.				
9.	ftpfileupload	It uploads a file to ftp server				Ftp server url username password file path to be upload

10.	generaterandomnumber	It generates a Random number and assign to variable. The limit of the random number is given in the 'Data Column'.				 VariableName (e.g. 1 25 XYZ)
11.	generateuniquestring	It generates the unique String and assign to variable. The length of the string to be generated is given in the 'Data Column'.				Length of string to be generated VariableName (e.g. 10 UserName)
12.	getobjectproperty	Get value of specified property of test object and assign it to a variable.	Need Parent	Need Object	Test	<property name> <variable name>

EXECUTION ENVIRONMENT

You can set 3 Environments for Krypton.

1. Any
2. Production
3. QA

Krypton have production.ini & QA.ini files in which you can set application-url, object timeout and some other parameters.

User can set Environment value in parameter.ini file.

Set Keyword : Production, QA and Any

Set Environment : QA/Production

You need to set Environment value in Test case file also. Under keyword column (Ie: Any, QA or Production). Only those test case will be executed which are categories as mentioned in parameteries.

If you don't have any concern about environment, set it as any.

Production.ini & QA.ini files must be there. No matter they are blank.

KRYPTON TEST CASES

Test Case sheet is a spreadsheet used to write test cases. This sheet should be in specific format to successfully run the test case. Below is the format of spreadsheet:

1. **Keyword (optional):** This field specify the execution environment of the test case.
2. **Test_scenario:** This field basically defines the Test scenario which one automating. For E.g.: "This Test case verifies login functionality".
3. **Test_case_id:** Each test case is given a unique Id. Format to define test case id is TC_TestCaseFileName.test_name (E.g. TC_SampleTest.test_case_01). TC_TestCaseFileName is test-case file name.
4. **Comments:** This field gives user a way to define each test step of a test case, these comments are visible in test reports and hence help users to debug the failed test case easily. For E.g.: If the test step includes entering name in the username field, one can write comment as "Enter name in username field" etc.
5. **Parent:** In this field user should give the name of the web page on which required object is present. User can give any name to the parent web page.
6. **Test_object:** It refers to the test object (Web element) present on web page. The combination of Parent and Test_object fields are used to uniquely identify test objects on the web pages.
7. **Step_action:** This field contains each step that is to be performed in a test case.
8. **Data:** This field work as a data provider to the step action like when entering username, actual username that is to be entered can come from this field.
9. **Iteration:** It works more like "loop" to take different values from test data sheet. For E.g.: specifying iteration- '1-10' will take values from row number 1-10 from the test data sheet.
10. **Options:** This field includes certain keywords such as: Optional {optional}, Ignore Case {ignorecase}, Partial Match {partmatch}, Ignore Space {ignorespace}, Skip/Ignore {skip}{ignore}. These keywords are optional but can be very useful in certain cases.

TEST DATA

The test_data sheet works as a data provider to the main test case file and should be named as "TD_TestCaseFileName". It includes the required input data for test case execution & column name, row # and the data which needs to be entered while execution.

Iteration: To execute a test case multiple times with different data, it would be handled by Iterations method. It can be distinguished with unique row numbers. For e.g. : if we'd like to enter Username as 'TestABC' & Password as 'test' then while execution in the iteration column of the test case sheet we would mention 1-1.

For e.g.: See the below sample for your reference

	A	B	C	D
1	row_no	Comments	Username	Password
2	1	Username and Password of site.	TestABC	test
3				

The Test Data sheet may include more than one sheet depending on test case requirements.

OBJECT REPOSITORY SHEET

Object Repository spreadsheet is used to store all web objects which are used in test cases. Below is the format and brief description of spreadsheet:

1. **sl_no:** It refers to Serial number.
2. **parent:** In this field user should give the name of the web page on which required object is present. User can give any name to the parent web page.
3. **test_object:** It refers to the test object (Web element) present on web page. The combination of Parent and Test_object fields are used to uniquely identify test objects on the web pages.
4. **logical_name:** As the name suggests it contains the description of test object which helps testers to identify objects.
5. **locale:** Language option in which the website under test is to be displayed. This field is optional and can be left blank.
6. **obj_type:** It contains information about type of web object. E.g. list, textbox, button, link etc.
7. **how:** In this section we define different methods using which web object can be found using selenium engine. For ex: xpath, css, id etc.
8. **what:** It contains the value of locator.
9. **comments:** This includes comments regarding the objects on a web page.
10. **mapping:** This refers to mapping the logical name of the object with the internal attribute.

REUSABLE TEST CASES

Reusable test cases are those test cases which are defined only once and can be used anywhere and any number of times. The best example of these test cases is Login test case.

For login we have to perform three steps each time i.e.

1. Enter username.
2. Enter password.
3. Click on login button.

Then why we should write all these each time whenever we have to perform login action?

Answer is we don't have to, if we use make login as a reusable test case.

It can be done as follows:

1. Write Test Case within same Test Case sheet
2. Create New Test Case file, containing all the reusable test cases.

Write Test Case within same Test Case sheet:

Create test case and use this within the sheet and whenever that test case needs to be executed just put **runexceltestcase** under "Step_action" followed by **Test Case Id**.

	A	B	C	D	E	F	G	H	I	J
1	keyword	test_scenario	test_case_id	Comments	Parent	test_object	step_action	data	Iteration	options
4										
5							runexceltestcase	TestCaseID		
6										

Example: For sending a message, reading emails and for more operations user need to login so we can write one single test case for login and use this test case where ever required.

Create New Test Case file:

This is similar to Test Case sheet, this sheet must be in specific format to successfully run the test cases. Format of spreadsheet is same as of Test Case sheet.

	A	B	C	D	E	F	G	H	I	J
1	keyword	test_scenario	test_case_id	Comments	Parent	test_object	step_action	data	Iteration	options
4										
5			TestCaseID				Login.001			
6										

How to use this feature:

1. Specify parameter ReusableDefaultFile value in parameter.ini file. (ie : ReusableDefaultFile: Methods.xls)
2. Create Test case file (spreadsheet) in specific format given above.
3. Specify sheet name (like: Login) and write test case according sheet name.
4. Test case Id must be according to sheet name. Ex : Login.001
5. At last specify test case to be called via another sheet under step action column.

KRYPTON PARAMETERS

1. **Browser:** Mention browser name on which execution is to done. (E.g. Firefox, chrome, IE)
2. **Driver:** Mention driver name (E.g. Selenium)
3. **TestCaseLocation:** Mention location of test-case excel sheet
4. **TestDataLocation:** Mention location of test-data excel sheet
5. **TestSuiteLocation:** Mention the location of test suite excel sheet
6. **TestCaseId:** Mention the Test case ID to be executed
7. **TestSuite:** Mention the Test suite name to be executed
8. **TestCaseFileExtension:** Mention the file format of test-case file (E.g. .xlsx, .xls)
9. **ReusableDefaultFile:** Mention the method (reusable) file.
10. **ReusableLocation:** Mention the location of reusable excel sheet.
11. **ORLocation:** Mention the path of Object Repository Excel sheet
12. **ORFileName:** Mention the name of Object Repository Excel sheet
13. **RecoverFromPopupLocation:** Mention location of pop-up Recovery Scenarios
14. **RecoverFromBrowserLocation:** Mention location of browser Recovery Scenarios
15. **RecoverFromPopupFileName:** Mention name of file which recover from pop-ups
16. **RecoverFromBrowserFileName:** Mention name of file which recover from browser
17. **LogFileName:** Mention Name of the file where the logs to be created
18. **LogDestinationFolder:** Mention the path where log/reports are to be generated
19. **ManagerType:** Mention the test management source systems (E.g. File System, MTM, XStudio)
20. **TimeFormat:** Mention the time format (E.g. : hh:mm:ss tt)
21. **DateFormat:** Mention the date format (E.g. : MM/dd/yyyy)
22. **DateTimeFormat:** Mention the date-time format (E.g. : dd/MM/yyyy HH:mm:ss)
23. **CompanyLogo:** Mention the full path with filename of the image to be used as the logo in report/log
24. **FailedCountForExit:** Refers to the maximum number of consecutive steps that can fail before the execution is terminated. Use 0 for no limit.
25. **EndExecutionWaitRequired:** Mention whether to wait or not for user response after execution of the test
26. **DBTestDataLocation:** Mention the location of Database Test data
27. **DBConnectionString:** Mention Database connection string if test need Database Access
28. **DefaultDb:** Mention the default db to work with (default is write mode)
29. **DBServer:** Default database server name
30. **DBQueryFilePath:** Mention the path of file in which database queries are stored
31. **DebugMode:** It highlights each and every step at run time. Its value can be given as true or false.
32. **ErrorCaptureAs:** It determines whether the page at which error has occurred should be captured in the form of an image or HTML file
33. **Environment:** Mention the environment of test execution
34. **EnvironmentSetupBatch:** Name of the batch file required for fulfilling certain pre-conditions before the execution of the test cases
35. **TestCaseIDSeparator:** Mention the test case id separator
36. **TestCaseIDParameter:** This parameter determines test case file name
37. **SnapshotOption:** Specify when to take snapshot
38. **RunRemoteExecution:** It determines whether the automation should run on a remote machine or not

39. **RunOnRemoteBrowserUrl:** Specifies the URL of the remote machine at which the execution should be carried out
40. **EmailNotification:** This parameter determines whether to send email notification or not
41. **EmailNotificationFrom:** This parameter determines the email id of the sender
42. **EmailSMTPServer:** It mention SMTP mail server (E.g. Smtplib@gmail.com)
43. **EmailSMTPPort:** It mention SMTP Port number
44. **EmailSMTPUsername:** Mention SMTP username for authentication
45. **EmailSMTPPassword:** Mention SMTP password for authentication.
46. **EmailStartTemplate:** This parameter mentions the location of start template file. This template is used to create email at the start of test
47. **EmailEndTemplate:** This parameter mentions the location of end template file. This template is used to create email at the end of test
48. **TestMode:** This parameter determines the environment name. Default environment is 'Control'. This is for A/B mode
49. **CloseBrowserOnCompletion:** This parameter determines whether to close the browser after execution of the test or not. Its value is given as true or false
50. **FirefoxProfilePath:** Mention the location of Firefox profile. Firefox saves user preferred information such as caches, passwords, certificate etc in a set of files called profile, which is stored in a separate location from the Firefox program files. User can have multiple Firefox profiles, each containing a separate set of information. User can use these saved profiles while executing the test cases, in order to execute the re-usable existing profiles. Apart from this, users have an option whether to use the existing profiles or would like to create a profile
51. **AddonsPath:** It determines the path where Firefox addons are saved
52. **ApplicationURL:** Mention the URL of web application you want to test. If URL is not given in data section of test-flow sheet then KRYPTON Engine use this parameter
53. **ObjectTimeout:** Mention waiting time in seconds for an object to appear on web page. If object is not present after the wait time completed, then it raise exception that object not found
54. **GlobalTimeout:** Mention the waiting time for all the actions to be performed before timeout
55. **MaxTimeoutForPageLoad:** It determines the maximum time required for a web page to load
56. **MinTimeoutForPageLoad:** It determines the minimum time required for a web page to load
57. **RecoveryCount:** Mention the number of times recovery can be done
58. **ListOfUniqueCharacters:** "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"