**The Selenium-IDE (Integrated Development Environment)** is an easy-to-use Firefox plug-in to develop Selenium test cases. It provides a Graphical User Interface for recording user actions using Firefox which is used to learn and use Selenium, but it can only be used with Firefox browser as other browsers are not supported. However, the recorded scripts can be converted into various programming languages supported by Selenium and the scripts can be executed on other browsers as well.

|  |  |
| --- | --- |
| **Title** | **Description** |
| Download Selenium IDE | This section deals with how to download and  Configure Selenium IDE. |
| Selenium IDE Features | This section deals with the features available  In Selenium IDE. |
| Creating Selenium IDE Tests | This section deals with how to create IDE  Tests using recording feature. |
| Selenium IDE Script Debugging | This section deals with debugging the Selenium IDE script. |
| Inserting Verification Points | This section describes how to insert  Verification points in Selenium IDE. |
| Selenium Pattern Matching | This section deals with how to work with regular expressions using IDE. |
| Selenium User Extensions | The Java script that allows users to customize  Or add new functionality. |
| Different Browser Execution | This section deals with how to execute Selenium IDE scripts on different browsers. |

**Installing the IDE** Using Firefox, rst, download the IDE from the Selenium HQ downloads page Firefox will protect you from installing add-on’s from unfamiliar locations, so you will need to click Allow to proceed with the installation, as shown in the following screenshot. Select Install Now. The Firefox Add-ons window pops up, rst showing a progress bar, and when the download is complete, displays the following. Restart Firefox. After Firefox reboots you will need the Selenium-IDE listed under the Firefox Tools menu.

**Opening the IDE** To run the Selenium-IDE, simply select it from the Firefox Tools menu. It opens as follows with an empty script-editing window and a menu for loading, or creating new test cases.

**IDE Features**

**Menu Bar**

The File menu has options for Test Case and Test Suite (suite of Test Cases). Using these you can add a new Test Case, open a Test Case, save a Test Case, and export Test Case in a language of your choice. You can also open the recent Test Case. All these options are also available for Test Suite. The Edit menu allows copy, paste, delete, undo, and select all operations for editing the commands in your test case. The Options menu allows the changing of settings. You can set the timeout value for certain commands, add user-defined user extensions to the base set of Selenium commands, and specify the format (language) used when saving your test cases. The Help menu is the standard Firefox Help menu; only one item on this menu UI-Element Documentation pertains to Selenium-IDE.

**Toolbar**

The toolbar contains buttons for controlling the execution of your test cases, including a step feature for debugging your test cases. The right-most button, the one with the red-dot, is the record button.

**Speed Control:** controls how fast your test case runs.

**Run All:** Runs the entire test suite when a test suite with multiple test cases is loaded.

**Run:** Runs the currently selected test. When only a single test is loaded this button and the Run All button have the same etc...

**Pause/Resume:** Allows stopping and re-starting of a running test case.

**Step:** Allows you to step through a test case by running it one command at a time. Use for debugging test cases.

**TestRunner Mode:** Allows you to run the test case in a browser loaded with the Selenium-Core TestRunner. The TestRunner is not commonly used now and is likely to be deprecated. This button is for evaluating test cases for back-wards compatibility with the TestRunner. Most users will probably not need this button. Apply Rollup Rules: This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action. Detailed documentation on rollup rules can be found in the UI-Element Documentation on the Help menu.

**Record:** Records the user’s browser actions.

**Test Case Pane** Your script is displayed in the test case pane. It has two tabs, one for displaying the command and their parameters in a readable table format. The other tab - Source displays the test case in the native format in which the le will be stored. By default, this is HTML although it can be changed to a programming language such as Java or C#, or a scripting language like Python. See the Options menu for details. The Source view also allows one to edit the test case in its raw form, including copy, cut and paste operations. The Command, Target, and Value entry display the currently selected command along with its parameters. These are entry where you can modify the currently selected command. The rst parameter specified for a command in the Reference tab of the bottom pane always goes in the Target element ld. If a second parameter is specified by the Reference tab, it always goes in the Value element ld. If you start typing in the Command, a drop-down list will be populated based on the rst characters you type; you can then select your desired command from the drop-down.

**Log/Reference/UI-Element/Rollup Pane**

The bottom pane is used for four different functions Log, Reference, UI- Element, and Rollup depending on which tab is selected.

**Log** When you run your test case, error messages and information messages showing the progress are displayed in this pane automatically, even if you do not select the Log tab. These messages are often useful for test case debugging. Notice the Clear button for clearing the Log. Also notice the Info button is a drop-down allowing selection of different levels of information to log.

**Reference** The Reference tab is the default selection whenever you are entering or modifying Selenese commands and parameters in Table mode. In Table mode, the Reference pane will display documentation on the current command. When entering or modifying commands, whether from Table or Source mode, it is critically important to ensure that the parameters specified in the Target and Value match those specified in the parameter list in the Reference pane. The number of parameters provided must match the number specified, the order of parameters provided must match the order specified, and the type of parameters provided must match the type specified. If there is a mismatch in any of these three areas, the command will not run correctly.

While the Reference tab is invaluable as a quick reference, it is still often necessary to consult the Selenium Reference document.

**UI-Element and Rollup**

Detailed information on these two panes (which cover advanced features) can be found in the UI-Element Documentation on the Help menu of Selenium-IDE. Selenium Commands Selenese

Selenium commands, often called selenese, are the set of commands that run your tests. A sequence of these commands is a test script. Here we explain those commands in detail, and we present the many choices you have in testing your web application when using Selenium. Selenium provides a rich set of commands for fully testing your web-app in virtually any way you can imagine. The command set is often called selenese. These commands essentially create a testing language.

In selenese, one can test the existence of UI elements based on their HTML tags, test for specific content, and test for broken links, input fields, selection list options, submitting forms, and table data among other things. In addition Selenium commands support testing of window size, mouse position, alerts, and Ajax functionality, pop up windows, event handling, and many other web-application features. The Command Reference lists all the available commands. A command tells Selenium what to do. Selenium commands come in three groups: Actions, Accessors, and Assertions.

**Actions** are commands that generally manipulate the state of the application. They do things like click this link and select that option. If an Action fails, or has an error, the execution of the current test is stopped. Many Actions can be called with the AndWait e.g. clickAndWait.

**Accessors** examine the state of the application and store the results in variables, e.g. storeTitle. They are also used to automatically generate Assertions.

**Assertions** are like Accessors, but they verify that the state of the application conforms to what is expected. Examples include make sure the page title is X and verify that this checkbox is checked.