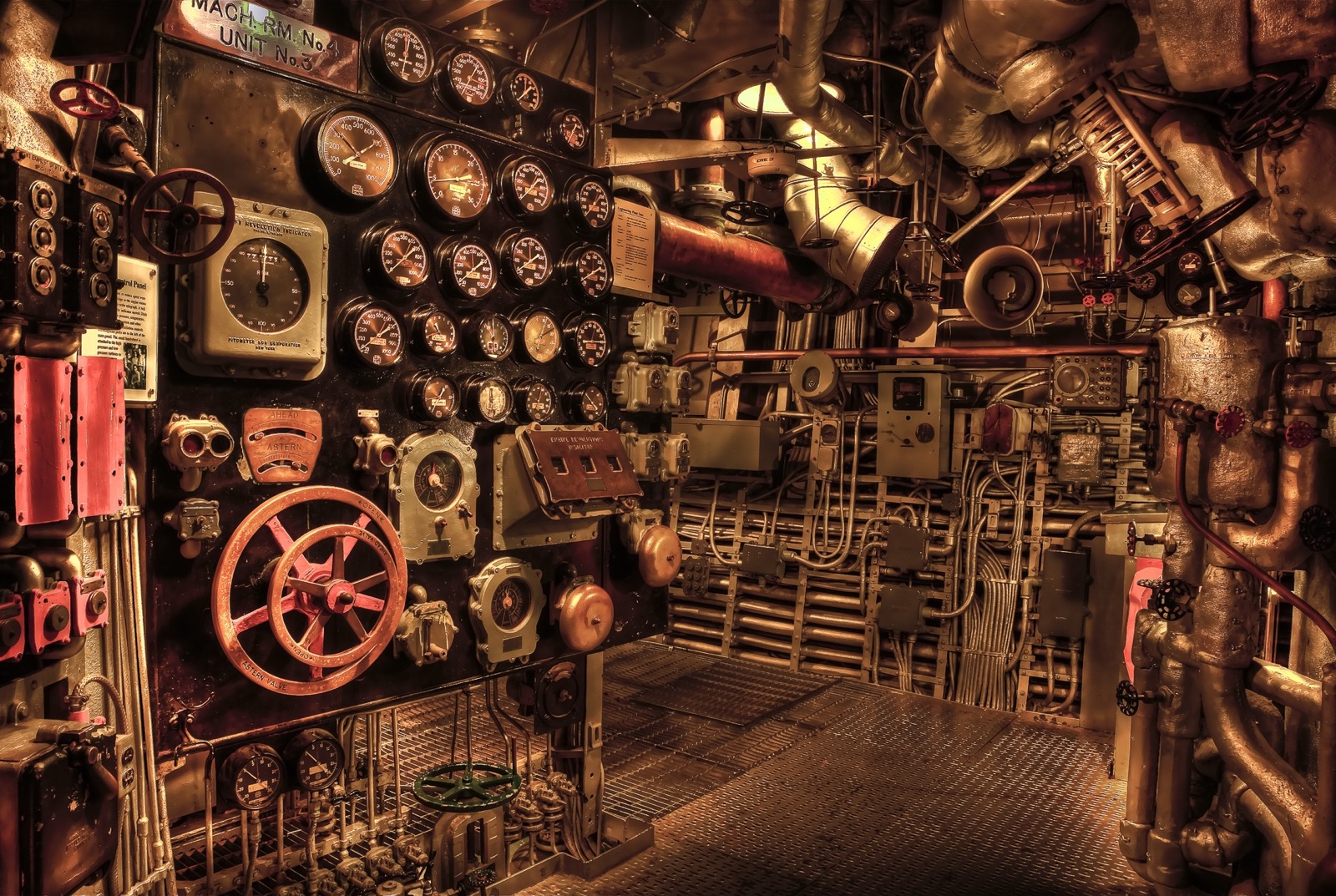
|  |  |  |
| --- | --- | --- |
|  |  | User Guide  Apex Test Automation Framework  Version: Beta 0.2 |

21 June 2017

#### Author: Simon Hunt

#### Location: https://github.com/schunt1/ataf/user\_guide.doc

#### Version: 0.1 Draft

****

“Testing Made Easy!“

# Contents

Contents 2

Introduction 3

Get Started 3

Document Notation 3

Abbreviations 4

Parameters 4

Projects 4

Create a Project 4

Update a Project 5

Delete a Project 5

Manage Users 5

Test Specification 5

Create a Test Specification 5

Edit a Test Specification 5

Delete a Test Specification 6

Duplicate a Test Specification 6

Test Cases 6

Create a Test Case 6

Update a Test Case 7

Delete a Test Case 7

Removing a Test Case 7

Managing Orphaned Test Cases 7

Test Conditions 7

Running a Test Case 8

Test Lab 8

Actions 8

Action Library 9

Bulk Copy 9

Selenium Script Commands 9

Simple examples 10

Complex Example 10

Substitution Strings 10

Test Data 11

Test Driven Design 12

Data Driven Testing 13

Continuous Integration 13

Reports 13

# Introduction

Welcome the Apex Test Automation Framework (ATAF), an Open Source project that uses Oracle Apex to build and manage Selenium IDE scripts. This user guide will walk you through the main components of ATAF as well is imparting some best practice tips and tricks for its use. The ERD in Figure 1 illustrates what ATAF is and how the various functional areas interact. The place to start is with the Test Condition (Fig 1, highlighted in red), which combines an Action, Page Item and Value to form an executable Selenium script. From there, the various levels of abstraction maximise code reuse and reduce the maintenance overhead of the automation scripts. The main thing that separates ATAF from other test frameworks is its ability to tap into the Apex metadata to determine a UI Map of the target application. Figure 2 illustrates how the ERD translates into the functional areas of the application.



Figure - ERD

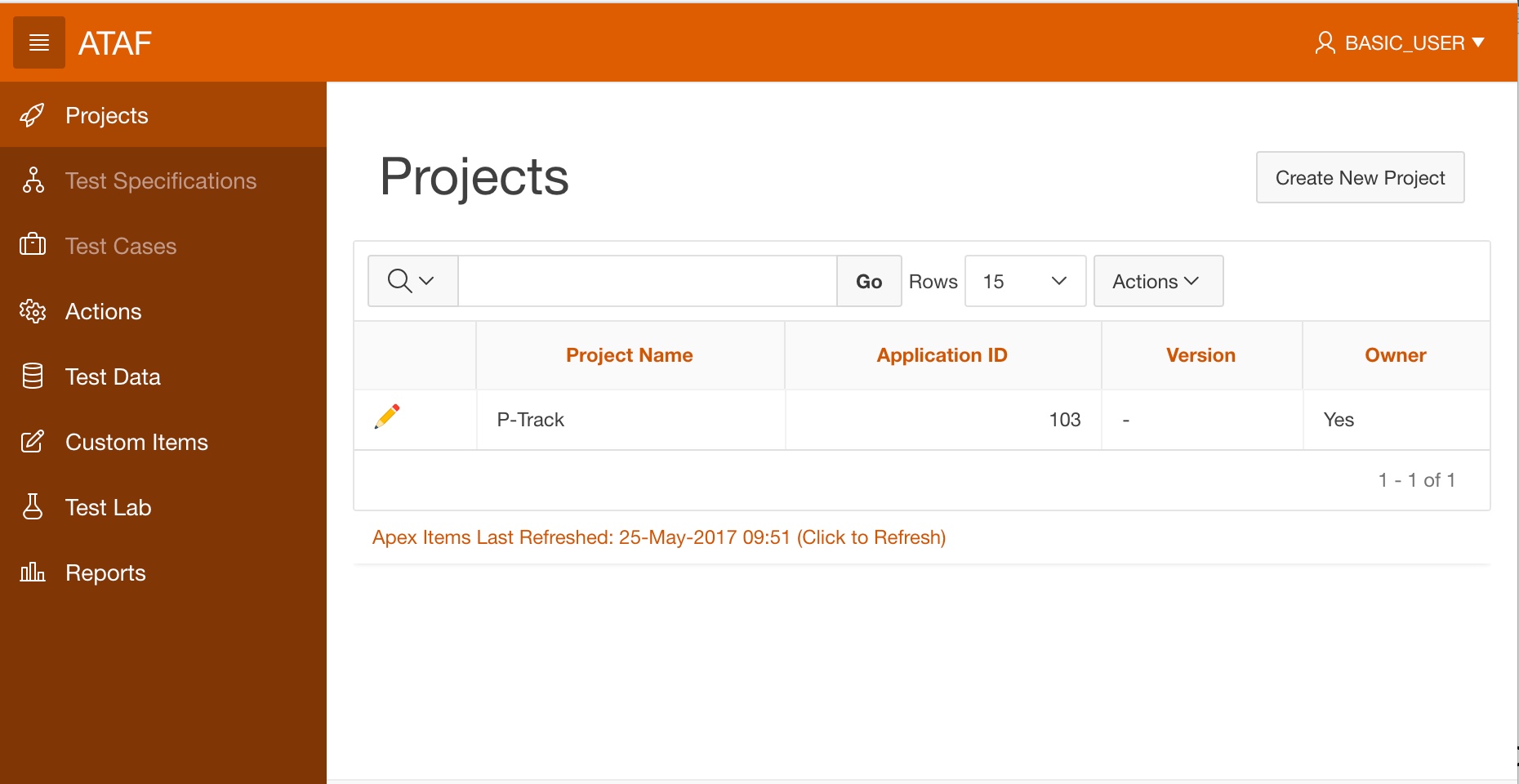


Figure – ATAF Screen Shot

## Get Started

A “get started” tutorial, which will introduce you to the basic components of ATAF is available at <https://www.apextestautomation.co.uk/get-started.html>

# Document Notation

The following notation and abbreviations will be used in this document:

* **Target Application**. The Target Application is the Apex application that is going to be tested by ATAF test scripts.
* **Function1/Function2**. The slash notation is used to represent a path through the application e.g. Reports/Test Case

## Abbreviations

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Description** | **Example** |
| ATAF | Apex Test Automation Framework |  |
| CI | Continuous Integration |  |
| TDD | Test Driven Design |  |
| DDT | Data Driven Testing |  |
| ERD | Entity Relationship Diagram |  |
| DAD | Database Access iDentifier | /ords/ |

## Parameters

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Description** | **Example** |
| #Project ID# | The ATAF Project ID, PK from Table ATAT\_PROJECT | 9714 |
| #ATAF Domain# | The Domain where ATAF is hosted | [www.apextestautomation.co.uk](http://www.apextestautomation.co.uk) |
| #ATAF Workspace# | Workspace containing ATAF Web Services | ataf |

# Projects

The project consists of a collection of Test Specifications, based on a specific Apex application and access can be restricted to particular users. Projects can use global components or have their own data sets, custom items and actions.

## Create a Project

Project creation is done from the Home page by clicking the Create Project button and entering the header information for the project.

* **Domain**. The Domain attribute is used to set the base domain for the Selenium script and for the #DOMAIN# substitution string used to build ATAF actions. When testing across multiple environments with different DAD’s, it’s easier to include the DAD in the Domain instead of in the action target string.
* **Version**. The target application version that the ATAF project is being written for.
* **Modules**. Modules are a useful grouping for Test Cases and would normally mirror Apex page groups. New modules can also be created when creating Test Cases.
* **Test Lab Username / Password**. From the account details page, copy the Selenium Grid Key and Secret, into the Test Lab Username and Password fields of the ATAF project.
* **Test Lab Suite ID**. Currently, users will need to manually link a Test Lab Suite with and ATAF Test Specification by entering the ID of the Suite into the Test Lab Suite ID field of the ATAF Project. Likewise, ATAF Specifications need to be linked to Test Lab tests. IE:
  + ATAF Project => Test Lab Suite
  + ATAF Specification => Test Lab Test

There are plans to automate the management of Test Lab suites and tests. In the meantime, this has to be done manually; but once done, the test steps are automatically synchronised each time the tests are run.

## Update a Project

Once a project is selected, it is possible to update the Project attributes by clicking “Edit Project” link in the Navigation bar.

## Delete a Project

Warning, deleting a project will remove all the project specific tests, data, and actions in one hit. In order to prevent the unintentional loss of a considerable amount of work, the user is prompted to type the name of the project prior to deletion. To delete a Project, select the project, click “Edit Project” and then “Delete”.

## Manage Users

Permissions for projects can be set to read or write. Select a project and then select “User Details” in the drop-down menu in the Nav Bar. The following options are available:

* **Open**. All users have read access and specified users can write.
* **Closed**. Only the Owner can access the Project.
* **Limited**. Individual users can be specified as have read or write access.

# Test Specification

The Test Specification is a grouping of Test Cases that when run in sequence will complete a particular test task e.g. function testing, end to end testing, regression test, load test etc. The Test Specification can be used for a complex scenario or simple function. In other frameworks and methodologies, the test specification may also be known as a Test Suite. Because specifications can be run in parallel, they should not have dependencies on other Test Specifications.

## Create a Test Specification

To create a Test Specification, edit a project and click the button Create Specification and complete the following fields:

* **Test Specification Name.** A standards based approach to naming the test specification will help other members of the team to understand what the test does. An appropriate naming convention could be: (Functional Area or Entity) and (Type or Sub Type) e.g. “Monthly Pay, End to End” or “Employees Data Load”.
* **Test Type.** The test types select-list is useful for categorising and sorting tests.
* **Automate.** The ‘automate’ checkbox, currently has no function other than to flag a particular test as being included in the Test Lab Test Suite. It is envisaged that this feature will evolve further.
* **Module**. Modules are the functional areas of the target application, which may correlate to Page Groups if they have been setup for the application. This provides a useful reference for where the tests are being targeted.
* **Order**. The order in which the test specification will display. Because test specifications can be run in parallel, it doesn’t represent the order the tests will run.
* **Notes**. A full description of what the test does and the expected outcome.
* **Test Lab Test ID**. The Test ID of the associated Test Lab test. In order to automate the test with the Test Lab, first create a blank test in the Test Lab, note the ID and add that to the Test Specification. Without this the Test Lab sync and run function will not work.

## Edit a Test Specification

To edit a Test Specification, first select it and then click the Edit Specification button in the top right hand side of the page.

## Delete a Test Specification

Deleting a Test Specification will NOT delete all the associate Test Cases, just the Test Specification. Any Test Cases not associated with other Test Specification’s will become orphaned and only viewable in Reports/Test Cases.

## Duplicate a Test Specification

The ability to duplicate a Test Specification is a useful timesaving function for creating a copy of specification and then adjusting it for another scenario. The Test Cases themselves are not duplicated, rather each Test Case will be assigned to both Test Specifications.

# Test Cases

The Test Case is the most important component of the Test Framework and the most difficult to get right. It has to be re-usable and adaptable in order to reduce the maintenance overhead the Test Specifications. By separating the Test Data from Test Case, it can be adapted for use in different scenarios.

Good Examples:

* Log In
* Log Out
* Create Project
* Edit Project
* Add Task

Bad Examples:

* Log in as an Administrator and Create a project
* Type text in the Last Text area on Page1 and submit the page.

Test Cases can set to cycle through all the records in a dataset, or linked to a specific record.

Example

|  |  |  |
| --- | --- | --- |
| Test Case | Data Set | Data Record |
| Log In | Users | Test User |
| Create Project | Projects | Default Data Record |
| Create Tasks | Tasks | Cycle Through Data |
| Delete Project | Projects | Default Data Record |
| Log Out |  |  |

## Create a Test Case

To create a Test Case, edit a Test Specification and click the Add Case. This will create a Test Case at the end of any Test Cases that exist for that specification. To Insert a Test Case, click the plus image in the row for the Test Case below the one you wish to insert. To insert a Test Case at the beginning, click the row’s plus image.

* **Test Case Name**. In order to promote Test Case reuse, when typing a Test Case name, you will automatically see a list of existing Test Case to choose from. If by the end of your typing no Test Cases are found, it will create a new one with that name. It is important to understand what Test Cases already exist for your project, a list of Test Cases can be found in the Reports section. Consistent name of Test Case will also make it easier to search for and locate existing Test Cases. Keep the Test Case high level and generic; use the associated data to make them specific. A suitable convention is to use *Verb - Function*, e.g. Create Person; where the verb is from a standard list.

|  |  |  |
| --- | --- | --- |
| Verb | Example | Notes |
| Create | *Create* Vehicle | Alternatives: Insert, Add |
| Read | *Read* Person | Alternatives: View, Open |
| Update | *Update* Project | Alternatives: Edit, Change |
| Delete | *Delete* Project |  |
| Log In |  |  |
| Log Out |  |  |
| Reset | *Reset* User Password |  |
| Remove | *Remove* Account | Soft Delete |
| Navigate To | *Navigate* Home | Useful for getting from any state to a known state |
| Copy | *Copy* |  |
| [other] | Reinstate Vehicle | Application specific functions |

* Sort Order
* Data Set
* Data Group

## Update a Test Case

To update a Test Case, click on the pencil icon in the interactive report. The Test Case attributes can be found in the hide/show region above the Test Conditions Interactive Report.

## Delete a Test Case

The Delete button permanently deletes the Test Case from the system and is not reversible. A confirmation message is displayed warning the user they are about to delete and permanently erase a Test Case.

## Removing a Test Case

In order to prevent accidental deletion of data, it’s vital to understand the difference between Delete and Remove. Delete will completely remove and permanently delete the test case including all other Specifications where it is used, whereas Remove just removes it from the current Test Specification.

## Managing Orphaned Test Cases

From time to time it is good housekeeping to review the unused Test Cases associated with a project and delete any orphaned (not used in a Specification) that are no longer valid. To this, go to Reports / Test Cases. All Test Case can be viewed along with a usage count and Edit function.

# Test Conditions

The test condition links an Action with a Page Item and Data Attribute e.g.

|  |  |  |
| --- | --- | --- |
| Action | Items | Data |
| Type into Text Field | P1\_PROJECT\_NAME | Project/Project Name |
| Select for Select List | P1\_PROJECT\_TYPE | Project/Project Type |
| Click Button | P1\_APPLY\_CHANGES |  |

When Creating a Test Condition each item is context sensitive which makes the feature easier to use and less error prone. First select the Page and the drop-down list of available actions will populate with only the actions that are possible on that page. Next select an Action and the list of page items will populate with only the valid items for that action on that page and so on. The test condition can use a Data Attribute from the data set linked to the Test Case, or any Data Attribute from another data set in the data library. Obviously, when using a different data attribute from the one associated with the Test Case it cannot be changed for different Test Scenarios and is there for constant.

**Outcomes**. Some Test Conditions may have a follow-on action that is triggered by the main action for example clicking a button may cause the page to branch or a modal window to open. It is not always obvious if an outcome is required, so the once the action is selected additional notes about that action can be viewed by clicking the icon at the end of the Action field.

Changing the order of Test Conditions. The order of Test Conditions can be changed by selecting a condition and change the order number in the advanced section.

## Running a Test Case

It’s useful to be able to run a test case in isolation whilst developing scripts in order to check it works as you expect it to. From the Test Case Interactive Report there is a column with a Selenium link that will generate a script, which can be downloaded and played. When generated from within a Specification, then it will use the context of that Specification to identify the data to use.

# Test Lab

The Test Lab feature allows ATAF to be used with a cloud hosted Selenium Grid, which offers the following benefits:

* Test Specifications can be run in parallel
* Tests can be run with different Operating Systems and Browsers
* Tests can be scheduled
* Tests can be automated as part of a Continuous Integration process

To enable this feature, you will first need to register with an account at:

<https://testingbot.com/>

# Actions

An ATAF Action is a smallest definable process in the framework and the lowest level of abstraction. The action can comprise of one or more browser action and when created as a group of browser actions can greatly simplify a complex process into a single action unit e.g. “Search an Interactive Report” or “Choose from Popup LOV”. The ATAF action links to Selenium actions but this could easily be adapted to use other browser automation scripting languages.

* **Action Name**. The name of Actions should be short, simple, consistent and “does what it says on the tin”. E.g. Click Button, Select from List, Type into Field etc.
* **Item Type**. The Apex Item Type that the action is associated with. This is used to set the context when building text conditions. IE if the action is linked to a button, only button items are selectable for that action. Where developers have created their own UI items, e.g. Hyper Links, these can be linked to Custom Items.
* **Script**. The field is used to produce a human readable version of the Test Script that can be particularly useful for documentation, training and user acceptance testing (UAT). Write an expanded version of the Action Name incorporating substitution strings that can be understood by a human viewing the UI. E.G. In the region '#REGION NAME#' click the '#LABEL#' button.
* **Project**. Actions can be Global or linked to a particular project. Only ATAF Administrators are able to create Global Actions and only users with write access to a project can edit a project’s actions. The idea being, the ATAF Administrator will manage the global action library, but if something special is required by a project they can add it themselves.
* **Notes**. The Notes are available to the Test when a particular action has been selected and can provide some useful information about how the action works, any limitations and the kinds of outcomes that are expected.
* **Test Data Required**. If the action requires any test data items, then this checkbox should be checked. If unchecked, then the data fields will not be displayed in the Test Condition.

## Action Library

Actions are linked to a Theme and the Action Library is a grouping of all tests for a particular Apex Theme. Actions are uniquely identified with a Global Unique Identifier (GUID) and derived Row Key for reference, so that Action Libraries can be exported and imported from one system to another.

## Bulk Copy

The Selenium Script Commands are attributed to an Apex Theme which is used during the generation of Selenium Scripts to identify which commands to use. When switching the themes of applications under test, it is important to ensure that there is an action library for the new theme. If one doesn’t exist then the “Bulk Copy” function on Action Home Page (p3) can be used to copy Selenium Commands from one theme to another. In many cases the commands will not change between themes, but identifying where they are different will soon become apparent when the scripts are run.

## Selenium Script Commands

The most technical challenge of ATAF is with building the Selenium Script Commands for Actions in the Action Library. The good news is that once a comprehensive library of actions has been created, then maintenance overhead is comparatively low.

* **Selenium Command**. The Scripting language used by selenium is Selenese, further information about the script syntax can be found at:

<http://www.seleniumhq.org/docs/02_selenium_ide.jsp#script-syntax>

The selenium commands supported by Testingbot can be found at: <https://testingbot.com/support/lab/commands.html>

* **Selenium Location**. Selenium supports a number of different locator attributes including the element id, class, name and link. By far the most reliable locator is the ID as this is guaranteed to be unique within Document Object Model (DOM). Where a standard locator is not appropriate, bespoke locators can be constructed using XPath. The XPath syntax is an extremely capable method of locating elements in the DOM, easily as good if not better, than the JQuery selector. However, the downside of XPath is that is designed for use with XML not HTML and therefore requires the HTML in the page to be as strongly formatted as an XML document. Therefore, if the HTML contains grammatical errors like not closing tags, the XPath selector may error. For further information on XPath:

<http://docs.oracle.com/javase/tutorial/jaxp/xslt/xpath.html>

<http://genius.com/2241980/Mat-brown-xpath-is-actually-pretty-useful-once-it-stops-being-confusing/Rfc-itself>

<https://addons.mozilla.org/en-us/firefox/addon/xpath-checker/>

<https://developer.mozilla.org/en-US/docs/Web/XPath/Axes>

* **Selenium Target / Apex Item Attribute**. If the location is a standard Apex Item Attribute e.g. the Name, ID, Label etc; then it can be selected from the Apex Item Attribute select list. If the location cannot be identified using a standard Apex Item Attribute, then the substitution strings can be used to create a custom target location. A range of substation strings are available to return values from the Apex views.
* **Test Data Value Field**. When the action is used in a Test Condition it may be linked to a data item. The Test Data Value Field identifies how that data item will be generated in the Selenium script. When set to No, there data item will not display in the Value field for that Selenium Command. Conversely, if set to Yes it will display. There are some special cases where other values from the substation list will need to be display in the value field e.g. Label = #Label#

## Simple examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action – Click Button** | | | | |
| Command | Location | Target | Attribute | Data |
| click | Id |  | DOM\_ID | No |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action – Type into Field** | | | | |
| Command | Location | Target | Attribute | Data |
| type | Id |  | Name | Yes |

## Complex Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action – Click Popup LOV** | | | | |
| Command | Location | Target | Attribute | Data |
| click | xpath | //table[@id='#NAME#\_holder']/tbody/tr/td[2]/a | Target | No |
| waitForPopUp |  |  |  |  |
| selectPopUp |  |  |  |  |
| type | id | SEARCH | Target | Yes |
| clickAndWait | xpath | //input[@value='Search'] | Target |  |
| click | xpath | //a[contains(text(), '#DATA#')] | Target |  |
| selectWindow |  |  |  |  |

A useful way of building actions is to use the Selenium IDE to record an action and then use the generated script as the start point for the action. The IDE will offer a range of different target alternatives for identifying items in the DOM.

## Substitution Strings

The substitution strings can be used to add attributes from the Apex Item attributes into the Selenium Target and Selenium Script fields. The values can be viewed by in the Apex Items report found in the reports section of ATAF. A link to the report can be found at the bottom of the Action Details page (p4). The substitution strings available are as follows:

#ID#

#DOM\_ID#

#NAME#

#LABEL#

#DATA#

#PAGE TITLE#

#ELEMENT#

#APP ID#

#PAGE ID#

#REGION ID#

#REGION NAME#

#ROW#

#DOMAIN#

#OUTCOME PAGE ID#

#OUTCOME PAGE TITLE#

# Test Data

ATAF Test Data consists of a Dataset with up to 20 data attributes and any number of rows of data. Each Attribute is created with a default value that can be used to build the Test Cases with. Additional rows of data can be added subsequently in order to change the Test Case or exercise the Test for a range of different data values. Each data row is named so that it can be referenced with the default value always being name “Default”. Data Sets are uniquely identified with a Global Unique Identifier (GUID) and have derived Row Key for reference, so that Data Sets can be exported and imported from one system to another.

* **Data Attribute Number**. The attribute number is not sort order and there is no constraint between the attribute and Test Condition, this is so that data sets and data rows can be interchanged between Test Cases. Note that changing the Attribute number once it has been used will have unexpected consequences for the Test Cases that use it.
* **Data Values**. The data value can either be explicit or derived from a function. A number of predefined functions have been created in ATAF for calculating dates, or generating random strings and numbers. Alternatively, you can set the value to be selected at random from one of the data rows by selecting “Random from the Bulk Data” in the function column.
* **Project**. ATAF ships with some generic data sets to get you started, which are Global. This means they are available for all projects and can only be managed by ATAF Administrators. Project specific datasets can be created by users that have write access to a project.
* **Adding new data functions**. Additional functions can be added by editing the apex application and adding a new display and return value to the static LOV used for the function column on page 8. Then add the function call to the case statement in the ATAF\_FULL\_TEST\_DATA\_V view. Please make sure that any configuration changes are recorded and checked following a patch or update.
* **Data Rows**. Data rows can be added and managed by clicking the Edit button in the Bulk Data region of the Test Data details on Page 8. Data can also be added from spreadsheets by clicking on the Bulk Load button in the same region. Note that when loading from a spreadsheet the data row must have a data row name. There are some useful internet services available for creating bulk datasets that can be loaded e.g.

[www.generatedata.com](http://www.generatedata.com)

[www.doogal.co.uk](http://www.doogal.co.uk)

* **Data Groups**. Data Groups are a great way to change Test Cases for different modes of testing e.g. Positive or Negative testing; or for use in different scenarios. Data Groups can be attributed to a Data Row, Test Case and Test Condition. The primary relationship is between the Test Condition and Test Data. If a Test Condition is allocated to a Data Group then it will only if that data row is in that group. If the Test Case is assigned to a Data Group, then only Data Rows in that Data Group will be used. The Group relationship between the components can be expressed as:

TEST CONDITION

TEST DATA

TEST CASE

The table below illustrates for each scenario whether the Test Condition will be included in the generated script or not.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Data  Row Group | Test Case  Group | Test Condition  Group | Outcome | Notes |
|  |  |  | Included | When no groups are used, all are included |
|  |  | Group X | Excluded | If the test case and test data have no groups then all conditions are included irrespective of Group |
|  | Group X | Group Y | Excluded | It can be useful to set a context for a test case so that only certain conditions are included for that test case. |
|  | Group X | Group X | Included |  |
|  | Group X |  | Included |  |
| Group X | Group Y |  | Excluded |  |
| Group X | Group Y | Group Y | Excluded |  |
| Group X | Group Y | Group X | Excluded |  |
| Group X | Group X |  | Included |  |
| Group X | Group X | Group Y | Excluded |  |
| Group X | Group X | Group X | Included |  |
| Group X |  | Group Y | Excluded |  |
| Group X |  | Group X | Included |  |
| Group X |  |  | Included |  |
| Group X |  | Group Y | Excluded |  |
| Group X |  | Group X | Included |  |

* **Custom Items**. One of the main components of a Test Condition are the Apex Items. However, Apex applications will often feature custom UI components that have been created by developers where the Apex framework is deficient. This could be in the form of an Apex Plugin, HTML output from a Custom PLSQL region or other HTML Mark-up embedded in the page. It is strongly recommended that ID attributes are added to any custom HTML items in order that they can be located by automated test tools.

# Test Driven Design

With Test Driven Design the Test Cases are written before the code is written and the code written until the tests are passed. This process helps to focus the development process on achieving the goal and ensures that tests are appropriate. In Agile terms, the tests can be derived from the acceptance criteria for the project backlog item. ATAF supports TDD by having the facility to add and assign Test Cases without creating the underlying Test Conditions. The process is further optimised by allowing users to define Test Data sets whilst creating Test Cases. The association of Test Data with a Test Case can be easily maintained using the Tabular Form controls on the Test Case IR.

# Data Driven Testing

ATAF utilises the power of the Apex Data Workshop to integrate test data with test scripts. Having created the scripts users can load spreadsheets of data and set the scripts to cycle through each line at a time. Also, built into ATAF, is the ability to generate its own data of the following types: string, number, date, alpha numeric; or it can be selected at random from a seed table loaded by the user. This means that Test Cases can be used many times in different scenarios by using a different data row. Further use of Data Groups will allow Test Cases to vary in nature based on the data being applied; this is particularly useful where data values will change the outcome of a Test Case.

# Continuous Integration

Continuous Integration is a process used in Agile development to regularly build, test and release code. ATAF scripts can be pulled from a development environment, via a web service, to a build server where they are run on a test environment. This process can be on-commit to a source control tool, on demand or for most Apex projects, run daily. Results are then emailed to the development team and displayed in a dashboard. ATAF integrates nicely with Selenium Server and build managers like Team City, Hudson, Jenkins and Team Foundation Server. The integration can be done via a database connection to your ATAF schema or via a web service call.

* **Database Connection**. If the CI Server can connect safely to the ATAF schema, then project scripts can be called using the following plsql:

ataf\_test\_lab.ci\_trigger(#project\_id#);

* **Web Service Call**. Where a DB connection between the CI environment and ATAF is not possible, then a web service call can be used to trigger the tests.

https://#ataf\_domain/ords/#ataf#/ci/#project\_id#

* **PowerShell Script**. Following is an example of how the web service can be called from a PowerShell script.

Invoke

-WebRequest

-usebasicparsing

-Method POST

-Uri https://#ataf\_domain#/ords/#ataf#/ci/#project\_id#

# Reports

The reports section primarily identifies which pages and items are covered by a test script and therefore supports a risk based approach to testing by ensuring that the important parts of the application are most thoroughly tested. Other reports identify which scripts will error following changes to Apex items as part of the normal development process. Finally, there is report that puts all the scripts into a human readable format that can be downloaded and included as part of the application document set. Following is a list of the reports currently available in ATAF.

* Page Coverage by Group
* Application Page Items
* Application Pages
* Items Tested
* Test Cases
* Test Case Errors
* Test Lab Results