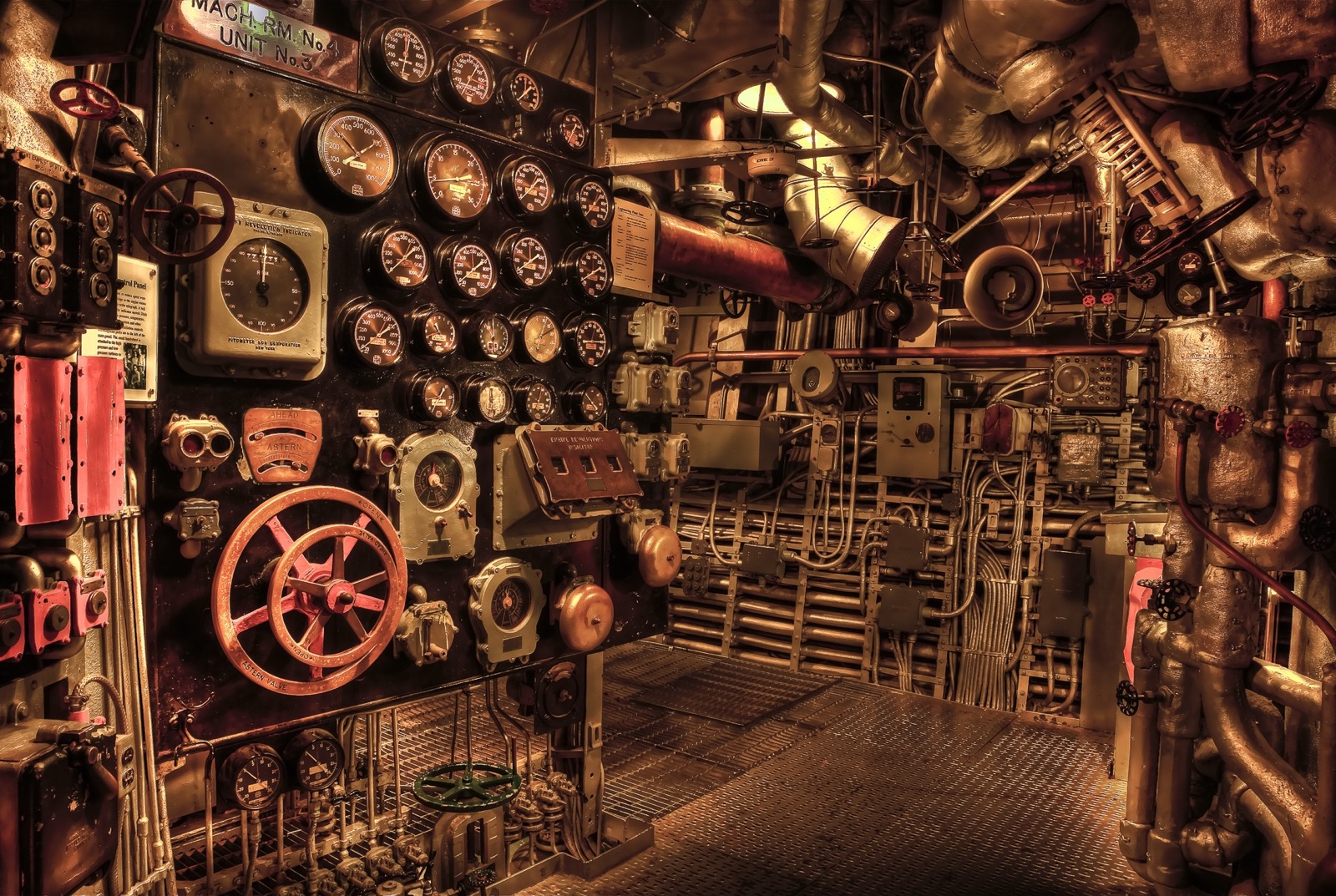
|  |  |  |
| --- | --- | --- |
|  |  | User Guide  APEX Test Automation Framework  Version: Beta 0.2 |

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 6

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 8

Create a Test Condition 8

Changing the order of Test Conditions. 9

Running a Test Case 9

Test Lab 9

Actions 9

Action Library 10

Bulk Copy 10

Selenium Script Commands 10

Simple examples 11

Complex Example 11

Substitution Strings 11

Test Data 12

Custom Items 13

Test Driven Design 14

Data Driven Testing 14

Continuous Integration 14

Reports 15

# 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

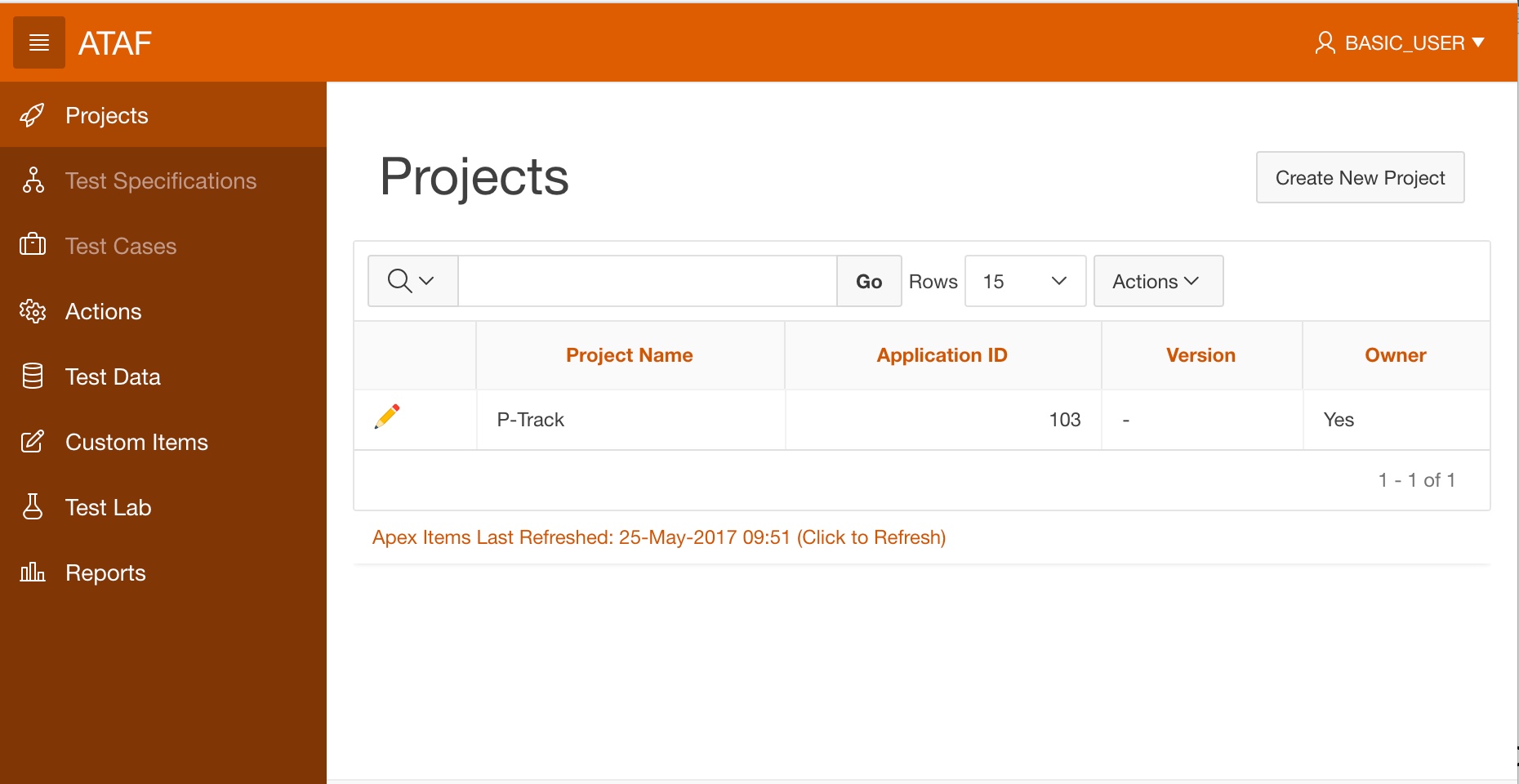


Figure – ATAF Screen Shot

## 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](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.
* **Heading > Sub Heading**. Represent a location in this document.

## 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/ |
| DOM | Document Object Model |  |
| AUT | Application Under Test | P-Track |
| GUID | Global Unique iDenentifier |  |
| XP | eXtreme Programming |  |

## 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 on the top right hand side of the page.

## Delete a Test Specification

Deleting a Test Specification will NOT delete all the associate Test Cases, only the Test Specification itself. As a result, Test Cases may become orphaned if they are not associated with any Test Specifications. Orphaned Test Cases are 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 Case’s 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 also, the most difficult to get right. It has to be re-usable and adaptable in order to reduce the maintenance overhead of automated testing. By separating the Test Data from Test Case, a single Test Case can be used in different scenarios. So keep your Test Cases simple and non-specific e.g.

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 beset to cycle through all the records in a dataset, or linked to a specific record e.g.

|  |  |  |
| --- | --- | --- |
| 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 | <no associated data set> |  |

## 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 the list of existing test cases. 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 very beginning, just click the first row’s plus image.

* **Test Case Name**. In order to promote Test Case reuse, when typing a Test Case name, you be presented a list of existing Test Case to choose from. If by the end of your typing, no Test Cases remain; it will create a new one with that name. So that you don’t duplicate Test Cases, it’s important to understand what Test Cases already exist for your project. To that end, a list of Test Cases can be found in the Reports section in the sidebar menu. Consistent name of Test Case will also make it easier to search for and locate existing Test Cases. Try and keep Test Case high level and generic; then 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**. The order in which the Test Cases are displayed and run. To change a Sort Order, edit the Test Case and change the order number in the Advanced section.
* **Data Set**. Specify the Data Set that is used for the Test Case. Once specified, the Test Condition can use a data attribute from the data set. This concept is vital for making you Test Cases reusable, which will reduce the maintenance overhead of automated testing.
* **Data Group**. It is possible to link a Test Case to a particular subset of data; or to run a particular path through a Test Case. For more information on this see the section below, [Test Data](#_Test_Data) > Data Groups.

## Update a Test Case

To update a Test Case, click on the pencil icon in the interactive report. The test case’s 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’s used, whereas Remove just removes it from the current Test Specification.

## Managing Orphaned Test Cases

From time to time it’s good housekeeping to review the unused test cases associated with a project and delete any that are orphaned or no longer valid. To this end, go to Reports/Test Cases where all the test cases can be viewed along with a usage count and an edit link.

# Test Conditions

Simply put, A Test Condition is the combination of an Action with a Page Item and a 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 |  |

## Create a Test Condition

From the Test Case click the button Create Condition. In the modal dialog, select the page and then a drop-down list of available actions will populate with only the actions that are relevant to 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. For Actions that require data, select a Data Attribute from the data set linked to the test case, or use a value from the global datasets. Note, when using a global data attribute with a Test Case, it cannot be changed for use with other test scenarios and is therefore less flexible.

* **Page**. This is the page on which the Test Condition will run. It is not currently possible to use a page alias; however, it is intended to add this as a feature.
* **Action**. There are two types of Action, global and project specific. The global actions are predefined and owned by the ATAF administrator, whereas project specific actions are editable by users with write access to the project. The list of actions displayed or only those relevant to the items on the page you’ve specified for the Test Condition; i.e. if there is a button on the page, you’ll only be given button related actions e.g. Click Button.
* **APEX Item**. A list of APEX items for the specified Test Condition. This is drawn from the APEX Views and any Custom Items that have been created for the project.
* **Data Options**. Not all actions required data, but for that do, here you are able specify whether to use an attribute from the data set associated with the Test Case or one from any other Data Set.
  + **Data Attributes**. Having selected to use an attribute from the associated dataset, you can select either an existing attribute or choose to create a new one.
  + **Data Item**. Alternatively, if you selected to use any data item, you can use the popup LOV to search for a data item in any dataset. This is equivalent to hardcoding data and will not provide the flexibility to change the test scenario with a different data row.
  + **Data Row**. Having chosen a Data Item, you are able to choose the row from which the Data Item will be taken. For example, you might choose the Data Item: Text String and the Data Row: < 50 chars.
* **Row**. The row item is used for Actions based on a report or tabular form, where you’d need to specify the row against which the Test Condition will run. There are values for the first and last row, as well specific row numbers. It is recommended to apply a report filter or sort in order to consistently position a row.
* **Outcome**. Some actions in a target application may have an associated follow on outcome. For example, clicking a button may cause the page to branch; or for a modal window to open. These are known as Outcomes in ATAF. It is not always obvious if an Outcome is associated with Action, so once an Action is selected some additional notes about the action are available by clicking the icon at the end of the Action field.
* **To Page**. The To Page, is the page where the application will go to when as a result of the Outcome.
* **Order**. The order in which the Test Condition will run. This will automatically be set by default, based on how and where the Test Condition is created from. If created using the Create button, the order will be set to end. If added from the report the order will be before the insert point.
* **Data Group**. If the Test Condition is linked to a Data Group, then it will only run if the test data set or Test Condition is set to that group.
* **Not in Group**. Is the reverse of the Data Group function except for null values, which are run.
* **Notes**. Useful notes about the Test Condition.

## Changing the order of Test Conditions.

The order of Test Conditions can be changed by selecting a condition and changing the order number in the Advanced section.

## Running a Test Case

whilst developing scripts, it’s useful to be able to run a Test Case in isolation to confirm it works as expected. In the Test Case Interactive Report there is a column called Download with a Selenium link that generates a script, which can be downloaded and played. If the Test Case is being accessed from within a specification, then it will use the context of that specification to for the data. If being accessed from the reports area, then it will use the default data values.

# 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 Web browsers
* Tests can be scheduled
* Tests can be automated as part of a Continuous Integration process

To enable this feature, you will need to register for an account at: <https://testingbot.com/>

# Actions

An ATAF Action comprises one or more Selenium actions and can greatly simplify a complex process into a single action e.g. “Search an Interactive Report” or “Choose from Popup LOV”. Currently, the ATAF actions only works for Selenium, but following the same principles, ATAF could easily be modified to use other browser automation scripting languages.

* **Action Name**. The Action name 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**. This is the APEX Item Type that the action is associated with. This is used to set the context when building Test Condition’s. I.e. if the Action is linked to a button, only button page items are selectable for that action list. Where developers have created their own UI items (e.g. hyperlinks), these can be created as Custom Items.
* **Script**. The Script field is used to produce a human readable version of the Test Script, which is 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 specific 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.
* **Notes**. The notes are made available when a particular action has been selected for a test and can provide some useful information about how the action works, its 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

The Action Library is a collection of Actions that work for the applications in your organisation. Because Actions are uniquely identified with a Global Unique Identifier (GUID), an Action Library could be exported and imported between environments using the APEX data workshop. Users can therefore share libraries.

## Bulk Copy

The Selenium script commands are linked to an APEX Theme and are used during the generation of Selenium Scripts to identify which commands to use. When switching the theme of an application, it is important to ensure that there is an action library for the new theme. If an Action Library for the new theme doesn’t exist, then the “Bulk Copy” function on the 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. Using the Selenium IDE to record an action in APEX can help determine the Selenium commands to use. 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 standard locator attributes including the element id, class, name and link. As the ID is guaranteed to be unique within the DOM, it is by far the fastest and most reliable locator. 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; it’s 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 if appropriate. The Test Data Value field identifies how that data item will be generated in the Selenium script. When set to “No”, then the data item will not display in the Value field for that Selenium command. Conversely, if set to “Yes” then it will display. There are some special cases where other values from the substitution 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 DOM attributes into the Selenium Target and Selenium Script fields. The actual attribute values can be viewed 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 is made up of datasets with up to 20 data attributes and any number of rows as you want. Each Attribute is created with a default value that can ensures that at least one values is there for building and testing Test Cases. Additional rows of data can be added which, when applied, change the nature of the Test Case. For easy reference, each data row has a name, with the default value always being name “Default”. Data Sets are uniquely identified with a GUID and have a derived Row Key for reference. Data Sets can be exported and imported from one system to another.

* **Data Attribute Number**. The attribute number is not used for sorting and there isn’t a constraint between Attribute and Test Condition, which means 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, 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. The function call should be added to the case statement in the ATAF\_FULL\_TEST\_DATA\_V view. Be careful to make sure that any configuration changes are recorded and managed properly.
* **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. To help generate data, there are some useful internet services available for creating bulk datasets, which can then be loaded into ATAF 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 vary Test Cases for different modes of testing e.g. positive or negative testing. They can also be used to completely change the path through a test for different scenarios. Data Groups can be attributed to a Data Row, Test Case and Test Condition. If a Test Condition is allocated to a Data Group then it will only run for data rows 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 is as follows:

TEST CONDITION

TEST DATA

TEST CASE

The table below illustrates for a variety of scenarios, 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

APEX applications will often feature custom UI components that have been created by developers. This could be in the form of an APEX Plugin, HTML output from a Custom PLSQL region or some other HTML Mark-up embedded in the page. Where these custom components need to be included in automation test, then they can be manually added to the list of APEX items through the ATAF as Custom Item’s feature. Custom Items have the following attributes:

* **Item Name**. As this is used for the #NAME# substitution string the ATAF Action, you need to think about which DOM attribute to use for the name in order for the Action to work. Normally it will be the Name attribute.
* **DOM ID**. The ID attribute for the DOM element.
* **Page ID**. The APEX page on which the item is displayed. Use page 0 if the item appears on all or multiple pages.
* **Display Sequence**. When APEX items are displayed in a select list, they are sorted by item type and display sequence. This allows you to control the position in which they are displayed.
* **Item Label**. This is used for the #LABEL# substitution of the Action.
* **Item Type**. This is key! If this item acts like an APEX item type, then by setting it to that type, then you can use all the existing Actions to control it. If it behaves differently, then set it to Custom and create new actions for it.

# Test Driven Design

TDD is a concept taken from XP where the Test Cases are written before the code and coding stops once 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 of 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 utilities 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 one a line at a time. Also built into ATAF, is the ability to randomly generate data in 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 merely by changing to 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 triggered by an on commit to a source control tool, on demand or scheduled. Results are then emailed to the development team and displayed in a dashboard. ATAF integrates nicely with Selenium Server and build management tools like Team City, Hudson, Jenkins and Team Foundation Server. The integration can be done via a database connection to your ATAF schema or over a web service call, as follows:

* **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 can be used to identify how much of the application is being tested and therefore supports a risk based approach to testing. Other reports identify which scripts are in error following changes to APEX items which often happens as part of the normal development process. Finally, there is report that puts all the scripts into a human readable format for download and inclusion 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