**SAP BPC Test Automation Framework (BTAF) Guidebook**

**4 March 2021**

**Prepared by BPC Automation**

Table of Contents

[**Document Control Information** 3](#_Toc68153959)

[**Document History** 3](#_Toc68153960)

[**1. Introduction** 4](#_Toc68153961)

[**2. Key Functionalities** 4](#_Toc68153962)

[**2.1 Driver Based Calculations** 4](#_Toc68153963)

[**2.1.1 What’s Covered** 4](#_Toc68153964)

[**2.1.2 Framework Design** 5](#_Toc68153965)

[**2.1.3 Execution Process** 5](#_Toc68153966)

[**2.1.4 Technical Design** 11](#_Toc68153967)

[**2.1.5 Template Details** 11](#_Toc68153968)

[**2.2 Roles Based Testing** 13](#_Toc68153969)

[**2.2.1 Framework Design & Tests** 13](#_Toc68153970)

[**2.2.2 Execution Process** 13](#_Toc68153971)

[**2.2.3 Technical Design** 16](#_Toc68153972)

[**2.2.4 Template Details** 17](#_Toc68153973)

[**2.3 Report Validations** 19](#_Toc68153974)

[**2.3.1 Process Flow Diagram** 19](#_Toc68153975)

[**2.3.2 Internal Validation Framework Components** 19](#_Toc68153976)

[**2.3.2.1 Data Intersection Model (and table definitions)** 20](#_Toc68153977)

[**2.3.2.2 Report Mappings – Mapping Master** 22](#_Toc68153978)

[**2.3.2.3 Report Mappings – Individual Mappings** 22](#_Toc68153979)

[**2.3.3 Framework Setup (Folder Structure)** 23](#_Toc68153980)

[**2.3.4 Execution Process** 24](#_Toc68153981)

[**2.4 UI for execution** 31](#_Toc68153982)

[**2.5 HTML Reporting** 31](#_Toc68153983)

[**3. Software Requirements & Code Updates** 32](#_Toc68153984)

[**3.1 Setup Environment Variables** 32](#_Toc68153985)

[**3.2 Test Setup** 32](#_Toc68153986)

[**3.3 Set the JDK in IntelliJ (Required only to work on source code)** 33](#_Toc68153987)

[**3.4 Importing BTAF from ADO (Required only to work on source code)** 33](#_Toc68153988)

[**4. Installation and Source** 34](#_Toc68153989)

[**4.1 Framework Location** 34](#_Toc68153990)

[**4.2 Code location in ADO** 34](#_Toc68153991)

[**5. Future Development** 34](#_Toc68153992)

[**5.1 Report Validation Redesign** 34](#_Toc68153993)

[**5.2 Driver calculation validation redesign** 35](#_Toc68153994)

[**5.3 Day in the Life (Future state operationalized view)** 36](#_Toc68153995)

[**5.3.1 BTAF Day In a Life View** 36](#_Toc68153996)

[**5.3.2 BTAF Day in a life – In Sprint View** 37](#_Toc68153997)

[**6.** **Troubleshooting** 38](#_Toc68153998)

[**6.1 Object Identification Issues** 38](#_Toc68153999)

[**6.2 Other Maintenance** 41](#_Toc68154000)

[**6.3** **Stack error when running report validations** 42](#_Toc68154001)

# **Document Control Information**

|  |  |
| --- | --- |
| Document Name: | SAP BPC Test Automation Framework (BTAF) Guidebook |
| Project Name: | Project Orion |
| Document Author: | BPC Automation |
| Document Version: | Draft |
| Date Released: |  |
| Electronic File Name: |  |

# **Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Additions/Modifications | Prepared/Revised By |
| 1.0 | 9/23/2020 | Initial Draft Version of BPC test automation framework Guidebook | BPC Automation |
| 2.0 | 3/4/2021 | Updated to reflect latest functionality | BPC Automation |
|  |  |  |  |
|  |  |  |  |

# **1. Introduction**

The key objective of the BPC test automation framework (BTAF) is to create a user-friendly tool to enable BPC testing to Shift-Left enabling increased efficiency, earlier defect resolution, and improved coverage. The goal of this guidebook is to explain the features of BTAF and the usage protocol for the same.

# **2. Key Functionalities**

The following table summarizes the functionalities developed as a part of BTAF

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Functionality** | **Description** |
| **1** | Driver Based Calculations | Calculations for Azure Forecast Objects |
| **2** | Roles Based Testing | Role folder, Role workbooks and workbook functionality validations |
| **3** | Report Validations | Validate reports between Input forms and Ad Hoc Backend Reports |
| **4** | UI for execution | UI to run the afore mentioned items |
| **5** | HTML Reporting | Report out results in HTML format |

## **2.1 Driver Based Calculations**

This requirement is to test the calculation on the following Azure Forecast Input form objects

* ACR\_MOM
* ACDS\_MOM
* ACR\_Pipeline
* ADS\_Pipeline
* MACC\_Pipeline

### **2.1.1 What’s Covered**

The following are the calculations that are covered for each of the input forms. The formulas are derived from the requirements in ADO for each of the input forms

|  |  |
| --- | --- |
| **Workbook Name** | **Calculation** |
| ACR\_MOM | * ACR$ * MOM% |
| ADS\_MOM |
| ACR\_Pipeline | * Conversion Rate Calculation |
| ADS\_Pipeline |
| MACC\_Pipeline |

### **2.1.2 Framework Design**

BTAF is divided into two main components. One of the components is the UI. The UI provides the user to provide the database they want to connect to and to provide the credentials for login to an environment, if applicable (OR) to SSO into the environment.

The second component is the key word driven test case flow sheet. The key word test case flow sheet allows users to be able to design individual test cases based on predefined keywords. This makes it easier for developers to use the framework with minimal training as the keywords mimic common user actions.

The keyword sheet is uploaded into the UI and then executes the flow based on the inputs previously provided by the user in the UI. The test case flow sheet consists of the individual test cases and the logical flow defined as Actions and corresponding parameters. The test case flow sheet also enables the user to call custom functions created to perform actions specific to a workbook. Each execution is considered an individual test suite and every keyword sheet has a series of “test cases” that validate a particular functionality.

### **2.1.3 Execution Process**

Launch “BTAF Executor” from the BTAF Framework Folder

Graphical user interface, application

Description automatically generated

Enter the test case name – Free text

Select Functional Testing

Graphical user interface, application, Word

Description automatically generated

Select the execution environment

Graphical user interface, application

Description automatically generated

Login Type – Select SSO OR Credential

SSO – When user does not need to enter their credentials

Credential – When user needs to explicitly specify a credential for login

Graphical user interface, application

Description automatically generated

Enter Username and Password, if choosing credential login

Graphical user interface

Description automatically generatedGraphical user interface, application, table

Description automatically generated

Specify the role type of the user role being logged in as

Depending on the role type the corresponding sheet in the keyword framework is loaded for execution

Graphical user interface, application

Description automatically generated

Click on Browse and select the keyword sheet to be loaded

Graphical user interface, application

Description automatically generated

Add to Test Run button is enabled once the workbook to execute is loaded.

Click on Add to Test Run to add the execution

Graphical user interface, application

Description automatically generated

Click on Execute Tests to begin execution

The Test Run selections screen shows a preview of the test to be executed

Below is the result of an execution

Graphical user interface

Description automatically generated

The results are stored in the following location

C:\Users\<User Name> \Documents\BTAF Framework\test-output

The filename is in the format <TestCaseName (as provided in the UI)>ExecutionTimestamp

### **2.1.4 Technical Design**

The following is the high-level flow of information for the execution

Diagram

Description automatically generated

### **2.1.5 Template Details**

The Test Case Flow Keyword sheet is maintained by the testers depending on the flows to be executed and the template is designed as follows

**TestCaseFlow sheet**

Graphical user interface, application, table

Description automatically generated

The “TestCaseFlow” sheet has 9 columns: “Run Flag”, “Test Case Description”, “Test Case ID”, “Action”, “Parameter 1”, “Parameter 2”, “Parameter 3”, “Parameter 4”, and “Parameter 5”.

* Run Flag
  + Set as ‘Y’ if user wants to execute the test cases
* Test Case Description
  + A brief description of the test case
* Test Case ID
  + Future Development – To update ADO of test case result
* Action
  + Keyword action
  + Refer to the Keywords table for all action keywords.
* Parameters
  + The required parameters for the action Keyword
  + Refer to the Keywords table for required parameters

The Tab names must map to the role type as present in the UI. Based on the selection made by the user in the UI for a particular test run, the corresponding tab is loaded and executed

Graphical user interface, application, table

Description automatically generated

The following are the keywords that are built into BTAF.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Keyword** | **Description** | **Parameter 1** | **Parameter 2** | **Parameter 3** | **Parameter 4** |
| **OpenWorkbook** | Opens Excel Workbook with the desired name | Name of Role Folder  [BPC - Business Operations (FBI)] | Name of the Folder the workbook is in  [YBPD\_FI\_AZFCST\_PLANNR\_INP\_0000] | Name of Excel Workbook  [Azure Forecast Input Workbook] |  |
| **RunVariant** | Selects and loads the desired Variant for the workbook | Name of the Variant  [Test\_US] |  |  |  |
| **SaveWorkbook** | Saves the workbook. Prerequisite for validations. | None |  |  |  |
| **VerifyAreaSelected** | Verifies the area input in parameter 4 matches the text of a cell in desired Excel worksheet | Name of Excel Workbook  [Azure Forecast Input Workbook] | Excel sheet name  [ACR\_MOM] | Cell Reference of cell to check  [B4] | Text to verify  [Area Selected: Australia] |
| **ValidateReport** | Validates a report and its corresponding input form | Name of the report workbook  [Azure Forecast ACR Report] | Name of the report worksheet name  [ACR Summary Report] |  |  |
| **WriteToCell** | Writes desired value into an Excel cell | Excel sheet name  [ACR\_MOM] | Cell reference  [A2] | Desired input  [Hello World!] |  |
| **VerifyACR\_MOM** | Validates the calculations of the ACR\_MOM or ADS\_MOM sheet | Name of the worksheet to verify  [ACR\_MOM] | Cell name in the top row of the table to validate  [Field Segment] | Cell name of the column to start the validation at  [JUL, 2020] |  |

**Note:** The flow of the keywords mimics a typical user execution. For any execution, the following keywords are more or less mandatory

* OpenWorkbook
* RunVariant
* SaveWorkbook

**Note:** Variants need to be created beforehand, depending on the selections to be made

**Note:** Each test case in the keyword sheet will relaunch a new instance of excel, and hence the actions will need to be ordered accordingly

## **2.2 Roles Based Testing**

Roles based testing involves the testing of security access for different user roles such as BizOps, Admin, Report and Planner.

### **2.2.1 Framework Design & Tests**

A separate input sheet has been designed to facilitate Roles based testing. The reason for the same is to ensure that roles-based testing can begin operating agnostic of BTAF development, while still remaining a part of the overall BTAF tool.

The following items are being tested as a part of roles based testing

* Access to specific folders as per requirements
* Access to specific workbooks as per requirements
* Ability for user to check access to Refresh, Calculate, Save and Submit as per the requirements
* Verify no error messages occur when user performs actions, they have access to

### **2.2.2 Execution Process**

Launch the “BTAF Executor” from the BTAF Framework folder

Graphical user interface, application

Description automatically generated

Enter the test case name and select “Roles Testing”

Graphical user interface, application

Description automatically generated

Load the Roles based test keyword sheet using the Browse button

Click on Add to Test Run once the button is enabled

Graphical user interface, application

Description automatically generated

Once the test is added to the test run selections, click on Execute tests to start execution

The below is an example of the report generated

A screenshot of a computer

Description automatically generated

Graphical user interface

Description automatically generated

The results are stored in the following location

C:\Users\<User Name> \Documents\BTAF Framework\test-output

The filename is in the format <TestCaseName>ExecutionTimestamp

### **2.2.3 Technical Design**

Below is the technical information flow for the Roles based test

Diagram

Description automatically generated

### **2.2.4 Template Details**

The Roles Based Test Case Flow Keyword sheet is maintained by the testers depending on the flows to be executed and the template is designed as follows

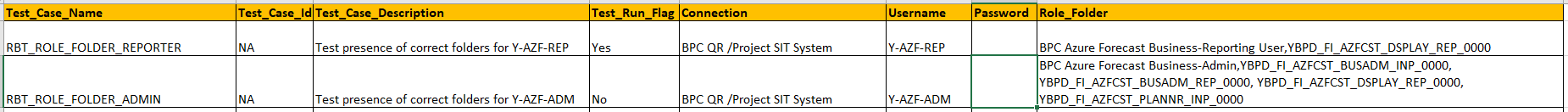
3 Tabs – Role Folder validations, Role Workbook Validations and Functionality validations

Graphical user interface, application

Description automatically generated

**Note:** The tabs must be present on all sheets to be able to execute Roles Based Tests

**Role Folder Tab template:**



Test\_Case\_Name: User defined test case name

Test\_Case\_ID: Future development to map to ADO test case

Test\_Case\_Description: Brief description of test case

Test\_Run\_Flag: Set as “Yes” if it needs to be executed

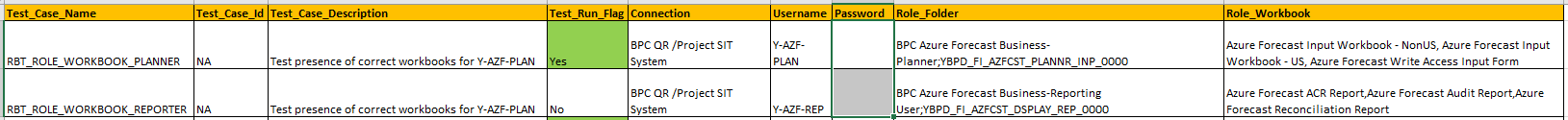
Connection: Select the connection

User Name: Specify the user name to login with

Password: Specify the password to login with

Role\_Folder: Specify the Role Folders specific to the user role, each folder comma separated

**Role Workbook Tab template:**



Test\_Case\_Name: User defined test case name

Test\_Case\_ID: Future development to map to ADO test case

Test\_Case\_Description: Brief description of test case

Test\_Run\_Flag: Set as “Yes” if it needs to be executed

Connection: Select the connection

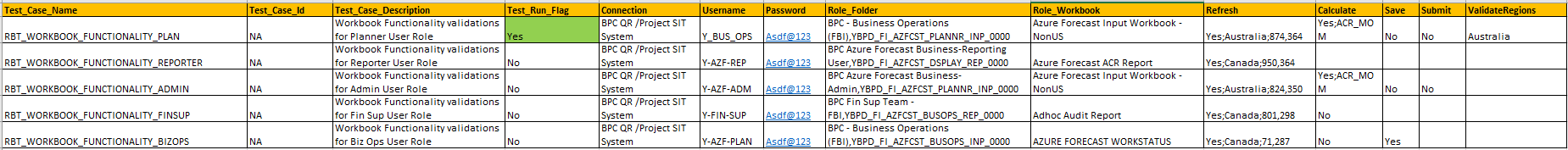
User Name: Specify the user name to login with

Password: Specify the password to login with

Role\_Folder: Specify the Role Folders specific to the user role, each folder comma separated

Role\_Workbook: Specify the Role Workbooks under a specific Role folder, comma separated

**Functionality test Tab template:**

****

Test\_Case\_Name: User defined test case name

Test\_Case\_ID: Future development to map to ADO test case

Test\_Case\_Description: Brief description of test case

Test\_Run\_Flag: Set as “Yes” if it needs to be executed

Connection: Select the connection

User Name: Specify the user name to login with

Password: Specify the password to login with

Role\_Folder: Specify the Role Folders specific to the user role, each folder comma separated

Role\_Workbook: Specify the Role Workbooks under a specific Role folder to be tested

Macro1: Specify the User defined name and the actual macro name in the format

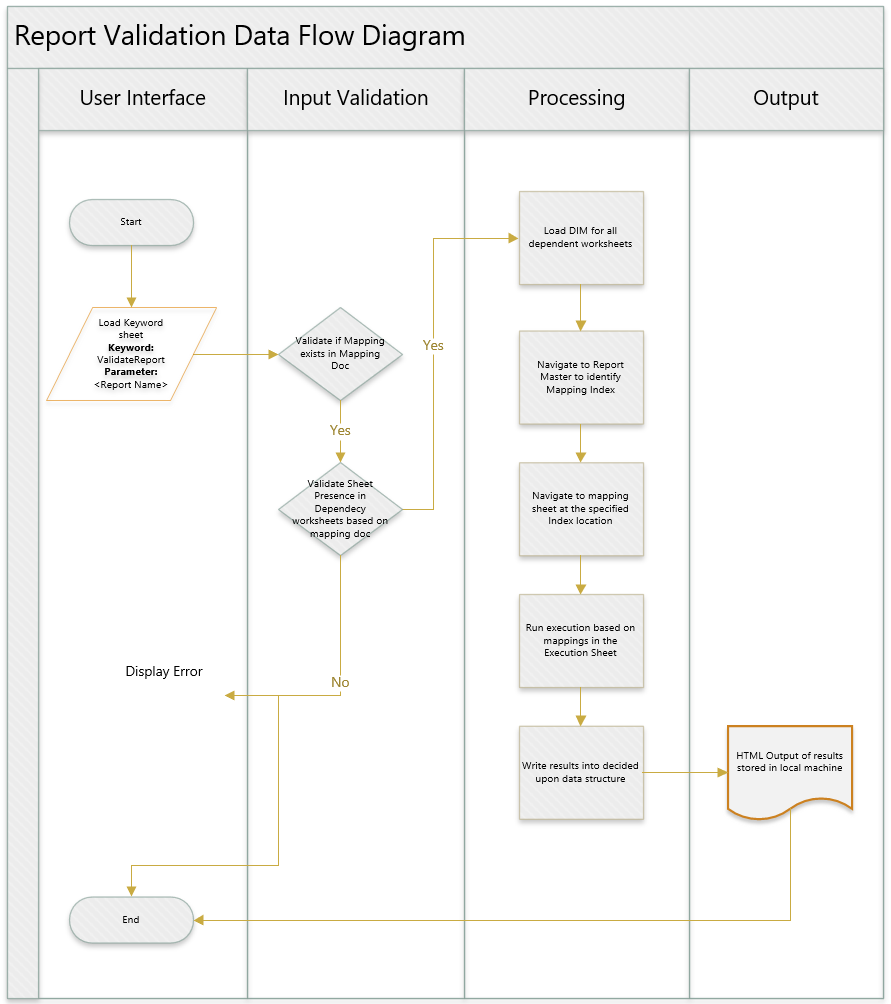
UserDefinedName;ActualMacroName

The next 5 columns allow the user to define the same for running specific macros

## **2.3 Report Validations**

The report validation mechanism has been implemented to validate the data on the individual reports against the Input Forms and backed, by way of ad hoc reports. As with the afore mentioned testing procedures the validations are driven from the UI using the keyword driven approach.

### **2.3.1 Process Flow Diagram**

The below is a process flow diagram as to how data flows from initiation of the test through the framework until the output

### **2.3.2 Internal Validation Framework Components**

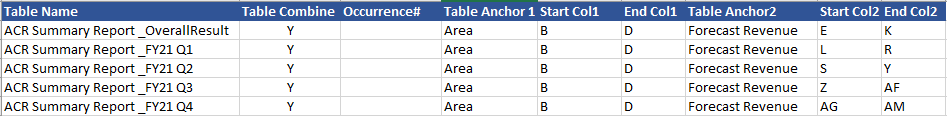
Three major components constitute the core of the report validations

* Data Intersection Model
* Report Mappings – Mapping Master
* Report Mapping - Individual Mappings

### **2.3.2.1 Data Intersection Model (and table definitions)**

The data intersection model is the primary basis for the report validations to function. The model is a representation of the data on the workbook divided into tables. Before diving into the DIM’s it is important to understand the table definitions

The table definitions define the structure of the workbook data in the form of tables. The table definitions are used to break down the workbook tables into individual user defined tables that will later be used in the DIM. The structure of the Table definition is as follows

****

The following are the usages of the columns in the table definition

Table Name: User defined to break down a workbook into login groups based on overall workbook design

Table Combine: Used specifically when Characteristics are separated into various KPI aggregations based on periods (For ex. ACR Summary Report)

Occurrence#: If a table anchor (See Below) is repeated, this can be used to specify the occurrence number (Ex. ACr comparison Report)

Table Anchor: The top left most text in a table that helps the framework to anchor to the starting cell of a table

Start Col: The starting column of the table being defined by the user

End Col: The ending column of the table being defined by the user

**Note:** To be able to use BODMAS type calculations, the intersection names must not have any spaces. For that reason, the table name cannot have spaces as in the screenshot above (“\_” may be used in lieu of the spaces)

The table definition will need to be defined as a part of the DIM for any workbook as below – WorksheetName\_TableDef

**Table

Description automatically generated with medium confidence**

Once the Table Def’s are defined, we can now define the DIM’s for the individual intersections.

The format for defining the DIM’s is as below

**Graphical user interface

Description automatically generated with medium confidence**

IntersectionName: User defined name for the intersection (In the example above the IntersectionName is created in the format using excel formulas

TableName|Value1|Value2|Value3|Select)

**Note:** All intersection names need to be unique and remove any special characters in the names to prevent execution errors (|,,, and \_ are OK to use)

**Note:** The mandatory requirement is that the intersection name starts with the TableName

Table: The table definition as defined in the tabledef sheet

Select: The Value to be selected

Column<n>: Generally the characteristic to be chosen

Value<n>: Generally the value

The framework will then construct it as a SQL Select clause as

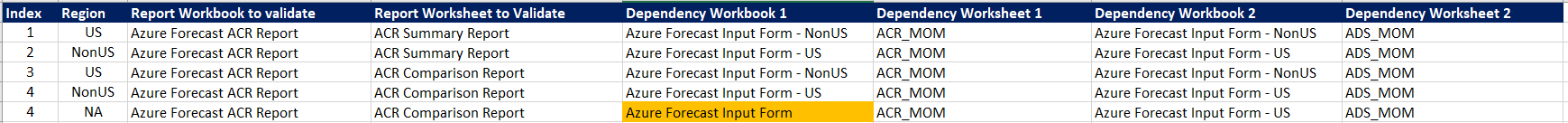
SELECT <Select> FROM <Table> WHERE <Column <n>>=<Value<n>>……

to extract the value from the table

**Note:** To be able to use BODMAS style calculations in the mapping, the intersection names cannot have spaces or special characters (except for the “|” delimiter). A formula or Replace all can be employed to replace the spaces with any other character (applied on column A) **.** As the intersection name is only a identifier to the query to be executed the user can follow any unique naming convention that takes care of the afore mentioned points.

### **2.3.2.2 Report Mappings – Mapping Master**

The mappings master is used to maintain the workbooks that contain the values required for validating the values in the report. This will also be used to maintain any adhoc reports that will be required to validate backend values.

****

Index: Defines the mapping sheet for a particular Report to Dependency Workbook Combination

Report Workbook to validate: The name of the workbook being validated

Report Worksheet to validate: The actual worksheet within the workbook that needs to be validated

Dependency Workbook <n> & Dependency Worksheet <n>: The workbooks containing the Expected values, present in the server

Based on the report to be validated and the workbooks present the appropriate index number is chosen for validation

### **2.3.2.3 Report Mappings – Individual Mappings**

All the mappings are to be maintained in the framework by the user for all the possible combinations as present in the Mapping Master. The individual mappings are maintained for a specific combination based on the Index number being named as the File Name of the mapping document.

For ex. If for a combination of X report worksheet and Y Dependency workbook, the index is 1, the mapping for this combination is named as “1”, enabling the framework to identify the mapping to be executed.

**A picture containing text

Description automatically generated**

Type: Specifies if the validation is a validation against Input Form or Backend Adhoc report

Report Intersection: The Intersection name being validated as mentioned in the DIM

Input Form Intersection: The intersection name that maps to the Report intersection in terms of the value

Ability is provided to perform basic mathematical functions to calculate the values

**Note:** The framework provides the ability to execute BODMAS style calculations.

For ex. If we need to perform a calculation similar to ((A-B)\*C)/C where A,B and C are intersection names, the same can be used similarly in the Input Form Intersection column to validate the respective report intersection

The resultant formula would look like ((Intersect A – Intersect B)\*Intersect C)/Intersect C similar to any BODMAS formula.

### **2.3.3 Framework Setup (Folder Structure)**

Within the BTAF Framework the following folders are to be present

* Data Intersection Models
* Report Mappings

**Table

Description automatically generated**

The DIMs are stored in the Data Intersection Models folder with each file named in the format <Workbook Name>;<Worksheet Name>

**Graphical user interface, text

Description automatically generated with medium confidence**

The Report Mappings folder contains the Report intersection Master and the individual mappings

**Graphical user interface, text, application

Description automatically generated**

The framework uses these folders and the appropriate files under them to run the tests.

### **2.3.4 Execution Process**

Launch “BTAF Executor” from the BTAF Framework Folder

Graphical user interface, application

Description automatically generated

Enter the test case name – Free text

Select Functional Testing

Graphical user interface, application, Word

Description automatically generated

Select the execution environment

Graphical user interface, application

Description automatically generated

Login Type – Select SSO OR Credential

SSO – When user does not need to enter their credentials

Credential – When user needs to explicitly specify a credential for login

Graphical user interface, application

Description automatically generated

Enter Username and Password, if choosing credential login

Graphical user interface

Description automatically generatedGraphical user interface, application, table

Description automatically generated

Specify the role type of the user role being logged in as

Depending on the role type the corresponding sheet in the keyword framework is loaded for execution

Graphical user interface, application

Description automatically generated

Click on Browse and select the keyword sheet to be loaded

Graphical user interface, application

Description automatically generated

Add to Test Run button is enabled once the workbook to execute is loaded.

Click on Add to Test Run to add the execution

Graphical user interface, application

Description automatically generated

Click on Execute Tests to begin execution

The Test Run selections screen shows a preview of the test to be executed

Graphical user interface, application, table, Excel

Description automatically generated

The generic flow of keywords for report validations is as below

OpenWorkbook: Opens workbook under a specified folder structure

RunVariant: Used to make selections on the prompts screen (Note: Variant will need to be created beforehand)

SaveWorkbook: Used to save the workbook to a temp location for validations

ValidateReport: Performs the validations as per the methodology defined in the previous sections

The results are stored in the following location

**Summary Report:**

C:\Users\<User Name> \Documents\BTAF Framework\test-output

The filename is in the format <TestCaseName>ExecutionTimestamp

**Detailed Report:**

C:\Users\v-vankas\Documents\BTAF Framework

The filename is <TestCaseName>ExecutionTimestamp

**Understanding the detailed report:**

Graphical user interface, text, application, email

Description automatically generated

The top right section of the report has the execution details such as the Test case name, the workbook and worksheet being validated and time details

The report has an execution summary which defines the summary of execution based on the table definitions on which the validations were performed

The report provides detailed execution for Input Form and Backend validation, based on the validation type defined in the Report Mapping sheet

On Expansion of this section, the results are broken down based on the table definitions validated and a summary of the results at that level

Graphical user interface, text, application, email

Description automatically generated

Upon further expansion, the detailed status at a validation level, based on the key figures validated is displayed. The results are displayed based on the validations from the Where clause mentioned in the DIM model in conjunction with the mappings being validated

Graphical user interface, application

Description automatically generated

## **2.4 UI for execution**

The UI that helps to perform the execution is developed using JavaFX. The UI is developed in generated from the source code housed in ADO in the following location.

[BAS-SFN-SAPPTP-BPC11E-TestAutomation - Repos (visualstudio.com)](https://microsoftit.visualstudio.com/DefaultCollection/OneITVSO/_git/BAS-SFN-SAPPTP-BPC11E-TestAutomation)

Once the changes, as necessary, are made to the code the project needs to be built for the changes to the UI and code to be pushed to the executable JAR file. This JAR file can be incorporated into the framework and published to sharepoint for the testers to use for testing. The remaining functionality of the UI remains BAU as mentioned in the previous sections.

## **2.5 HTML Reporting**

HTML reports have been implemented for the various validations (Functional/ Roles based). A customized report has been developed for the report validations to provide detailed results of the execution on top of a summary report. The locations of these reports are mentioned in the respective sections of each document.

# **3. Software Requirements & Code Updates**

The following are the key software requirements for BTAF

* Maven 3.6.1 (<https://maven.apache.org/download.cgi>) (Required only for development)
* IntelliJ Community version (<https://www.jetbrains.com/idea/download/#section=windows>) (Required only for development)
  + Other Java IDEs are acceptable, however no instructions or guidance available for using them
* JDK (<https://www.oracle.com/technetwork/java/javase/downloads/index.html>)
  + Recommended version: JDK 11.0.10

## **3.1 Setup Environment Variables**

* Open “File Explorer” --> Right click “This PC” --> Click “Properties” --> Click “Advanced System

Settings” --> Confirm elevation with credentials --> Click “Environment Variables”

* Add the following to your Path system variable if not present already (Left click “Path” --> click

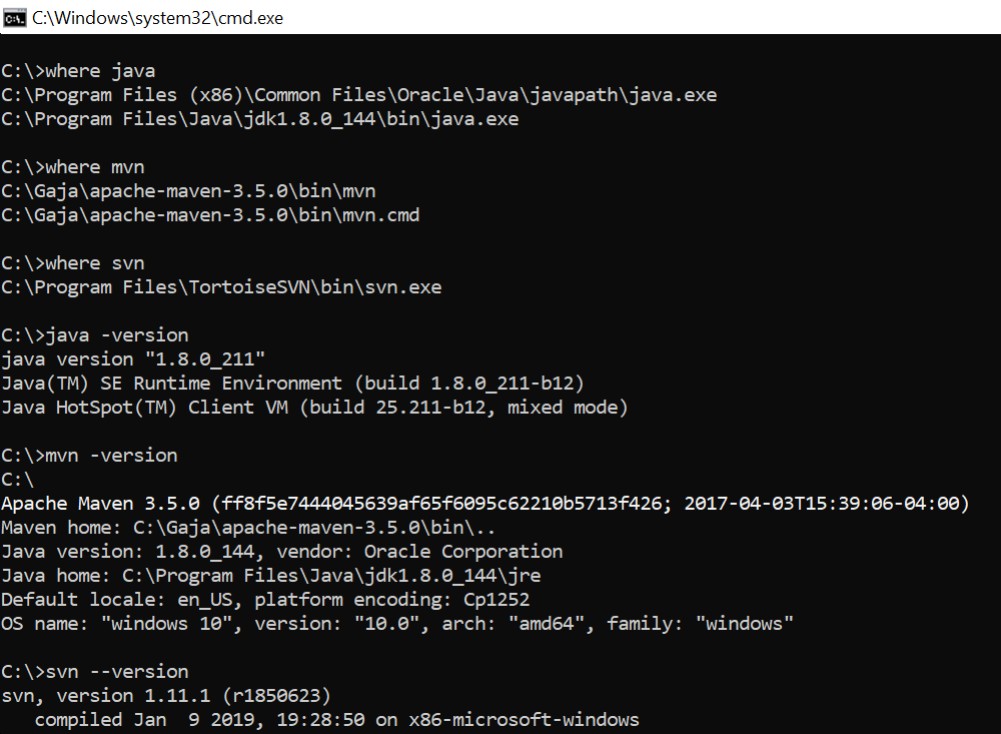
“Edit”)

* + Java Bin folder – e.g: C:\Program Files\Java\jdk-11.0.10\bin
* Setup new system variables (Click “New”)
  + JAVA\_HOME: <path to your JDK folder containing bin> - e.g C:\Program
  + Files\Java\jdk-11.0.10
  + CLASSPATH: %JAVA\_HOME%\lib\tools.jar (as is)

## **3.2 Test Setup**

Go to command prompt and type the following commands to verify the installation and versions are

all correct.



## **3.3 Set the JDK in IntelliJ (Required only to work on source code)**

* 1. Open IntelliJ.
* 2. Open the Project Structure dialog (e.g. Ctrl+Shift+Alt+S).
* 3. In the leftmost pane, under Platform Settings, click SDKs.
* 4. Click the green + icon and add the Java SDK path we set above

Graphical user interface, application

Description automatically generated

* 5. Click OK in the Project Structure dialog.

## **3.4 Importing BTAF from ADO (Required only to work on source code)**

1. Open IntelliJ  Click “File”  “New”  “Project from Version Control”
2. Select “Azure DevOps Git” from the Version Control dropdown
3. Click “Clone”
4. Sign-in to Azure DevOps Services
5. Select “BAS-SFN-SAPPTP-BPC11E-TestAutomation”  Click “Clone”

# **4. Installation and Source**

This section outlines the technical elements of BTAF and how to use them.

## **4.1 Framework Location**

The latest version of the UI is located in the below sharepoint path

* [Project Orion - Test\_automation - All Documents (sharepoint.com)](https://microsoft.sharepoint.com/teams/ProjectOrion2/Shared%20Documents/Forms/AllItems.aspx?id=%2Fteams%2FProjectOrion2%2FShared%20Documents%2F4%20%2D%20Build%2FTest%5Fautomation&p=true&originalPath=aHR0cHM6Ly9taWNyb3NvZnQuc2hhcmVwb2ludC5jb20vOmY6L3QvUHJvamVjdE9yaW9uMi9FZzluTHRpR3htRkVoM2ZzZWxGdWRPWUJHSVRnclJWdjVkUHlabUJoOVZtYnNnP3J0aW1lPThaem42Yl9EMkVn)

Download the BTAF Framework folder and save it in **Documents**

**Note:** The code must be moved to this location on the local machine - C:\Users\<User ID>\Documents and NOT the one drive documents location

## **4.2 Code location in ADO**

The source code for the project is present in the master branch under this repository in ADO

* [BAS-SFN-SAPPTP-BPC11E-TestAutomation - Repos (visualstudio.com)](https://microsoftit.visualstudio.com/DefaultCollection/OneITVSO/_git/BAS-SFN-SAPPTP-BPC11E-TestAutomation)

Refer to section 3.4 for cloning the code to the IntelliJ IDE to work on the source code

# **5. Future Development**

## **5.1 Report Validation Redesign**

The updated report validation redesign improves upon the existing design to utilize query definitions defined in SAP to query the database and compare the values against the values in the report, thereby eliminating the data intersection model and simplifying the configuration process.

Below is a diagrammatic representation of the high level flow

Diagram

Description automatically generated

The detailed validation mechanism is as follows with a couple of options to accomplish the same

Diagram

Description automatically generated

## **5.2 Driver calculation validation redesign**

The driver calculation redesign simplifies the configuration by employing the approach of using existing production results against regression results for validations.

**Graphical user interface, text, application, Word

Description automatically generated**

## **5.3 Day in the Life (Future state operationalized view)**

### **5.3.1 BTAF Day In a Life View**

The BTAF Day In a Life view is designed to provide a futuristic view of how BTAF integrates into the BPC testing lifecycle. Once functionalities are built into BTAF, the day in a life view provides a holistic view of activities and actors involved BPC testing process.

A picture containing timeline

Description automatically generated

Once the requirements are uploaded to ADO, the BTAF team will perform a gap analysis of the requirements and decide the net new functionality for which BTAF components will need to be developed.

The BTAF team will then absorb the development of these components in the current sprint, track development in ADO, then integrate the same into the BTAF repository and test the functionality within the cycle.

In the Near term, the BTAF team will impart training to the SWE’s to be able to execute the test cases/ test suites on their own and also be able to make changes to the test cases.

### **5.3.2 BTAF Day in a life – In Sprint View**

The following is deep dive view of the In-Sprint working of how BTAF development would work in the as the tool matures and components are built in to make it more robust and less maintenance intensive.

Timeline

Description automatically generated

While the end goal of the tool is to ensure maximum test coverage within sprints, there is possibility of certain functionalities/ components that cannot be developed and tested within a sprint. The above image seeks to address the same and provide a figurative view of the process that will be followed in these cases.

The BTAF test team, as mentioned in the day in the life view, will perform a gap analysis of the incoming functionality to determine net new functionality to be developed. The following are 3 logical flows resulting out of the analysis

**Functionality exists in BTAF:** If the functionality exists in BTAF, the BTAF team/ SWE team will execute the functionality and provide the results

**Functionality does not exist in BTAF:** The BTAF team will determine if the functionality can be developed AND executed in the current sprint.

If Yes, the functionality will be executed within sprint. If No, the functionality will move to a n-1 cadence where if a similar functionality exists in future sprints, it will be taken up for execution. While this is not actual execution, the BTAF team will ensure that the functionality is developed as per requirements and obtain the necessary approvals before integrating the code into BTAF

**Functionality that cannot be automated:** In some cases, there might be functionality that cannot be automated due to low reusability, technical limitations etc., which will need to be executed manually

# **Troubleshooting**

## **6.1 Object Identification Issues**

The framework uses mmarquee which is built on MSFT Windows automation to identify and interact with the physical excel objects such as AO edit boxes, buttons etc. There are times when object property changes can cause the interactions to fail due to change in object properties such as software upgrades and the like.

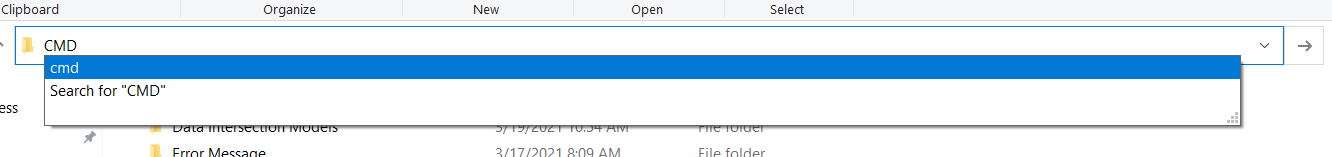
When BTAF is not able to identify a particular object an error similar to the one below will be encountered

Graphical user interface, text, application, email

Description automatically generated

In order to fix this issue it is important to identify where in the code the error is occurring. To identify the same launch BTAF using the following steps in debug mode

1. Navigate to the following location C:\Users\<User Name> \Documents\BTAF Framework
2. Type CMD in the explorer bar and click Enter



1. In the Command Prompt window that pops up, type java -jar BTAF.jar and click enter to launch BTAF

Text

Description automatically generated

1. Run the execution as you normally would until you see the error again, at which point stop the execution
2. Go to command prompt to identify the error thrown, which must be similar to the error shown above

Graphical user interface, text

Description automatically generated

1. To fix the issue, the first step will be to identify the actual line number as to where the error is occurring. This can be identified by reading through the stack trace and identifying the first line which corresponds to the our code.

Text

Description automatically generated

This can be identified from the actual BTAF code (refer to section 3.4 for more details) in IntelliJ which corresponds to the user defined classes. For ex. In the screenshot above the highlighted line corresponds to the following location in the code

Text

Description automatically generated

1. As the error is in Line 260, we will now need to update the properties of the object in that line namely mConnectionListView

Based on where the error occurred during execution, in this case the connection list, we will need to identify the property that changed.

As present in the code, getListByAutomationID is used to identify the list item using it’s automation ID. The automation ID is generally the unique identifier used to identify the object.

In order to confirm and update the same, we will be using the Accessibility Insights tool to identify the same

Graphical user interface, application

Description automatically generated

Point on any object in the connection list and you can identify it’s properties as below

Graphical user interface, text, application, table

Description automatically generated

As in the screenshot above, the AutomationID property is now “\_connectionListView”, whereas in the code it was “mConnectionListView”

This will need to be updated in the code to account for the change in property

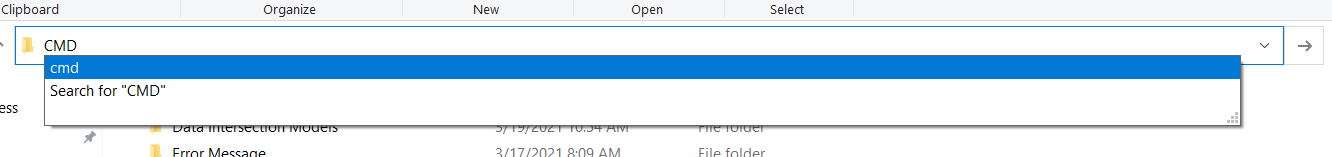


Now, on re running the execution should execute successfully. This will be the process to be performed in case of object identification issues

## **6.2 Other Maintenance**

To perform other maintenance on the code, follow the below steps to debug and fix the code

1. Navigate to the following location C:\Users\<User Name> \Documents\BTAF Framework
2. Type CMD in the explorer bar and click Enter



1. In the Command Prompt window that pops up, type java -jar BTAF.jar and click enter to launch BTAF

Text

Description automatically generated

1. Run the execution as you normally would until you see the error again, at which point stop the execution
2. Go to command prompt to identify the error thrown, which must be similar to the error shown above

Graphical user interface, text

Description automatically generated

1. To fix the issue, the first step will be to identify the actual line number as to where the error is occurring. This can be identified by reading through the stack trace and identifying the first line which corresponds to the our code.

Text

Description automatically generated

This can be identified from the actual BTAF code (refer to section 3.4 for more details) in IntelliJ which corresponds to the user defined classes. For ex. In the screenshot above the highlighted line corresponds to the following location in the code

Text

Description automatically generated

The user the can then begin work on debugging and fixing the issue, as required.

## **Stack error when running report validations**

If the execution is failing due to a stack error (overflow or empty) the most likely cause is there are some special characters that might not have been handled, that are present in the Report Mapping. In order to handle these, navigate to the following methods and update the special characters that need to be handled

* + PerformValidation.java -> checkForSpecialCharacters() and
  + PerformValidation.java -> evaluate()- line# 243

The special characters currently handled are &, \*, +,/, |,(,). Any other special characters will need to be removed, unless until the above methods are modified.