Runtime Library Manager
CBTA 3.0 SP11 - Customization
# Table of Contents

1 Foreword ........................................................................................................................................... 5

1.1 Terminology .................................................................................................................................. 5

2 Runtime Library Manager .................................................................................................................. 6

2.1 Starting the Runtime Library Manager (RTL) ................................................................................... 6

2.2 Customizing the Runtime Library .................................................................................................... 10

2.3 Creating a Custom Function ............................................................................................................. 13

2.3.1 Creating Custom Functions Manually ......................................................................................... 13

2.3.2 Creating Custom Functions with the Code Assistant ................................................................. 14

2.3.3 Submitting the Customization ..................................................................................................... 22

3 Appendix ........................................................................................................................................... 26

3.1 Displaying Custom Functions in the MIME Repository ..................................................................... 26

3.2 Known Limitations ............................................................................................................................. 27

4 References ......................................................................................................................................... 28

4.1 Materials and Documentation .......................................................................................................... 28

4.2 SAP Notes ....................................................................................................................................... 28
1 Foreword

The runtime library manager is a client-side tool to customize the VB script libraries that CBTA uses at runtime when executing test scripts.

It can write additional custom functions that the test scripts may need when automating tests of business scenarios for which the usual approach (based on default components) is not sufficient.

1.1 Terminology

Runtime Library
When executing a test script, the VB script coding for the test is sent to the client computer and executed using the VB script interpreter. The runtime library is a set of VB scripts which provide helper classes, functions and procedures to simulate actions that are normally performed by a regular user.

Default Components
Default components are components which perform atomic operations against UI elements. The runtime library contains default components for all UI technologies supported by CBTA.

GUI Scripting API
When writing custom functions for SAP GUI applications, as the test engineer, you may need to use the SAP GUI Scripting API, which provides interfaces to the application content.

Query API
When writing custom functions for web applications, as the test engineer, you may need access to the HTML content that the application displays in Internet Explorer. This can be done using the query API that is documented separately.
2 Runtime Library Manager

Prerequisites

- You have authorization for the RTL manager.
- You have enabled the RTL manager.

For more information about the prerequisites, see SAP Note 1912801.

2.1 Starting the Runtime Library Manager

1. In SAP Solution Manager (transaction SM_WORKCENTER), select the Test Suite -> Administration tile.

   ![Test Suite - Administration Tile](image)

   Figure 1: Test Suite - Administration Tile
2. In the new window, select the CBTA tab and choose the Customize RTL link.

![Customize RTL Link](image)

Figure 2: Customize RTL Link

3. In the RTL manager logon screen, enter your credentials. Depending on the system configuration you may have the option to logon using basic authentication or using client certificates. Note that the current user is selected by default and this cannot be changed.

![RTL Manager logon screen](image)

Figure 3: RTL Manager logon screen
4. You are connected to the SAP Solution Manager system in which the CBTA runtime library is stored.

You go to the RTL initial screen. The following information is displayed:

- the SAP Solution Manager system ID and client
- details of the version of the official runtime library that SAP delivers
- details about the customization (if any)

Figure 4: RTL Manager Initial Screen
Authentication Issues

It may happen that the authentication fails.

- Refer to SAP Note 1912801
- Ask the administrator of the system whether the CBTA_RTL_MANAGER service is properly configured.
- Ask the administrator for the required permissions

![Figure 5: RTL Manager - Authentication Issues](image)

Log and Traces

If the problem persists, check for the logs and traces. They are located at:

- %appdata%\Roaming\SAP\CBTA\logs\TAF_RTLManager.log
2.2 Customizing the Runtime Library

To create a new custom function to address test automation issues, download the runtime library from the SAP Solution Manager system, and make it available on the local file system of your machine.

Note

When the RTL manager starts for the first time, no customization is active, so the execution of a CBTA test script has previously relied on the official runtime library that SAP delivers.

1. Choose Open for Edit.

2. Specify where to store the runtime library on the local file system:
   1. Choose the Browse... button, to select an existing folder.
   2. Choose Ok to proceed.

   An additional panel appears. It shows
• the new state: Customization is pending
• the location selected

3. To navigate to the folder in which the custom code is to be stored, choose Explore.

You find the following data:

• The root folder `CBTA_Customization` contains a ZIP file containing the official runtime library (`CBASE.zip`).

• The `CBASE` folder contains the CBASE configuration and the VB script files (`*.vbs`). The content of the `CBASE.zip` file is extracted automatically.
Do not modify the CBASE configuration and the VB script files. Only VB script files stored in the Custom folder are to be modified.

- The sub-folder Custom contains the custom code that has been downloaded from the SAP Solution Manager system.

![Figure 9: Content of the CBASE folder](image)

**Note**

The first time, this folder is empty (or even missing). It is created automatically when clicking the Explore pushbutton.
2.3 Creating a Custom Function

You can create a custom code function either manually or with the code assistant.

2.3.1 Creating Custom Functions Manually

1. Create the custom VB script file (or modify an existing one). You can use any text editor to write the code.
   Ensure the following:
   - The VB script files have the extension `.vbs`.
   - The library has a meaningful enough name to avoid conflict with other libraries.
   - The code does not declare global variables.
   - Function and variable names start with a customer-specific prefix.

2. Store your custom VB script files in the `Custom` folder (`C:\CBTA_Customization\CBASE\Custom`, for example).
   The custom VB script files are displayed on the `Functions` tab page. If the `Custom` folder is empty, the `Functions` tab shows an empty list (Figure 10).

3. To check the syntax, start the VB script file manually, by double-clicking the file.
4. Choose `Refresh`, to update the list.
2.3.2 Creating Custom Functions with the Code Assistant

The code assistant is a wizard that allows you to select a pattern and generate a “ready for use” custom function.

1. Choose Code Assistant.
2. Choose Select....
   A dialog box appears. The patterns available are classified into categories, according to the UI technologies (Figure 11).
3. Select a pattern that matches the UI technology of the application being tested, and confirm.

   Example

   Select the SelectTransactionType pattern to start creating a custom function using queries for testing SAP CRM business scenarios.

4. Choose Next.
   The Code Assistant asks for the name of the new function (Figure 12). The default value comes from the pattern that you have selected.
5. Enter the function name.

Note

The code assistant does not allow two functions in the same library. The library name cannot be changed. Its value is updated with a default value automatically while typing the function name. The code assistant does not override existing functions (or existing libraries). It checks whether the library exists. In case of a naming conflict, it asks for a new name.

6. Choose Next.

Verifications are performed to make sure the library name is unique before generating the VB script file.

7. Choose Finish to complete the creation of the library (Figure 13).

The custom library is generated.
The newly created function is shown in the list.

8. To edit the function and edit the VB script file in a text editor, select it and choose Edit (Figure 14).

![Figure 14: Editing the Custom Library](image)

9. Adapt the generated script to your needs.
   In our example (Figure 15),
   - the name of the custom function already takes into account the function name specified while using the code assistant.
   - the recommended exception handling is already in place.
   - the code of the various patterns available gives concrete examples of how to use the different APIs available.
   In our example, the CRM query API accesses the content that the CRM application displays in the browser.

   Note

   *For more information, see the query CBTA - Test Automation - Query API documentation*
Until now the custom functions is marked as “Pending” in the list (Figure 16). This informs the test engineer that the other testers cannot yet see and use the changes. The customization is only available locally, and only the test engineer who opened the runtime library for editing, can test it.

10. To test the function, create a CBTA test script (from the Test Composition Environment) and invoke it explicitly.
Example: CRM Business Scenario - Create Sales Order

Our `XXX_SelectTransactionType` function searches for a transaction type in the list displayed by CRM when creating a sales order.

This typical CRM scenario is sensitive to the content of the database. When recording such a scenario, the generated test script includes all the user actions that the end user performs while searching for the transaction type, including the mouse clicks performed to navigate to the next pages when the transaction type is not shown on the first page.

The generated test script may run properly when executed immediately after being recorded, because the transaction type may be displayed in the same position, but new transaction types can be created and inserted into the database at any time. The test script may, for instance, select another transaction type, the one that is now displayed at the recorded position, and this leads to an unpredictable behavior.

1. Select the transaction type
   In this example, we select the transaction type `TA` (with the description `Telesales`) wherever it is.

   ![Figure 17: Selecting the Transaction Type](image)

2. Create the CBTA Test Script.
   Let's continue our example by creating a CBTA test script to test the SAP CRM Create Sales Order scenario.

   To create the CBTA test script, you require the following information:
   - System data container and target component pointing to a CRM system
   - Executable type: `CRM_WEBCLIENT`
   - Executable: `saprole=SALESRO&crm-logical-link=SLS-SLO-CR`
This application starts with the “Select Transaction Type” popup.

Figure 18: CBTA Test Script

Figure 19 shows the steps recorded by selecting the transaction type TA.

Figure 19: Recorded Steps

3. Modify the CBTA test script.

The test script must be adapted to be less sensitive to the database content. In our example (Figure 20), the recorded steps have been commented out by setting their state to Disabled.

Instead of these steps, we want to use our custom function, to search for the transaction type TA with the description Telesales. The fourth step has been added to illustrate how to do so with CBTA_CRM_A_INVOKE_FUNCTION.

The input parameters of the CBTA_CRM_A_INVOKE_FUNCTION component are the following:
- **Library** ➔ `Custom\XXX_SelectTransactionType.vbs`
- **FunctionName** ➔ `XXX_SelectTransactionType`
- **Parameters #1 to #4, and the Options parameters, are passed to the custom function.**

![CBTA Test: ZCR_CREATE_SALES_ORDER](image)

**Figure 20: Invoking a Custom Function**

4. Execute the CBTA test script.

   As the test engineer, execute the CBTA test script to check the custom code (no syntax errors) and that the implementation performs the expected operation.

   In the execution report (Figure 21), we can see that the step had to go to page #2 and page #3 to find the TA transaction type on row #45.
The test behavior is consistent; our custom function has now been tested.
2.3.3 Submitting the Customization

To make the test available to all test engineers, and make it reusable in other test scripts, submit the customization, by uploading it to the SAP Solution Manager system.

Prerequisites

You have tested the custom function.

Procedure

1. On the Customization tab page, choose Submit.

In Figure 22, the customization state is still Pending. On choosing Submit, you go to the Submit Pending Changes dialog box (Figure 23).
2. Enter the company name and the version of the customization. These values are not verified. They are only used internally to generate a "customization signature".

3. Optional: Select a change request and the package.

Note

The change request and the package must have been created previously on the Solution Manager System via transaction SE80 and SE09.
If you leave the change request empty, the customization is submitted as a local object. No package is required in that case).

Select a change request if the customer landscape runs several SAP Solution Manager systems. In such a landscape, you may want to transport the customization from the current SAP Solution Manager system to another system.

![Select Change Request Dialog](image)

**Figure 25: Select Change Request Dialog**

4. To submit the Customization, choose OK (Figure 23).

   The customization state is set to Active (Figure 28), so all test engineers can now run the test script, and check the behavior of our new custom function.

![Transport Organizer: Requests](image)

**Figure 26: Verifying the Submission in Transaction SE09**

The runtime library manager keeps track of the company name, the version, the user who did the customization, and the date.
The content deployed on the local file system is not deleted. The customization code is still visible (in our example under C:\CBTA_Customization\CBASE\Custom).

However, when executing the CBTA test scripts, the libraries stored locally are not used anymore. The customization is now stored centrally on the SAP Solution Manager system, and its content is deployed automatically (and on-demand) on each machine running CBTA.
3 Appendix

3.1 Displaying Custom Functions in the MIME Repository

The CBTA runtime library is stored centrally in the MIME Repository of the SAP Solution Manager system. To check the content of the MIME Repository, do the following:

1. Call transaction SE80.
2. In the MIME repository, choose SAP -> PUBLIC -> CBTA (Figure 28).

This folder contains the official runtime library (CBASE.zip) that SAP delivers. As shown in Figure 28, additional files are stored at that location when submitting the customization.

![Figure 28: Content of the MIME Repository](image-url)
3.2 Known Limitations

Runtime Library Manager prior to CBTA 3.0 SP02

The runtime library manager described in this document exists since the version 3.0 SP02 of CBTA (which requires Solution Manager 7.1 SP10 at least).

With previous CBTA releases, the alternative to the runtime library manager was to use the CBTA_RTL_MANAGER report, which is now deprecated.

Concurrent Check Out

There is no locking mechanism in the backend, so several test engineers may customize the runtime library at the same time, and conflicts may occur when submitting the changes. The runtime library manager does not provide any tool to solve these potential conflicts.

Versioning of the Runtime Library Customization

There is no versioning mechanism in the MIME repository, so always backup the custom libraries.
4 References

4.1 Materials and Documentation

Some additional documents have to be considered to take benefit of the Query API.

<table>
<thead>
<tr>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBTA – Runtime Library &amp; Default Components</td>
</tr>
<tr>
<td>CBTA – Custom Code Patterns</td>
</tr>
<tr>
<td>CBTA – Query API</td>
</tr>
<tr>
<td>CBTA – Test Recorder</td>
</tr>
<tr>
<td>CBTA – Object Spy</td>
</tr>
</tbody>
</table>

4.2 SAP Notes

SAP Notes mentioned in this document:

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1778899</td>
<td>CBTA - Collective Note</td>
</tr>
<tr>
<td>1912801</td>
<td>CBTA – RTL Manager – Runtime Library Customization</td>
</tr>
</tbody>
</table>