# Table of Contents

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

1.1 Terminology ........................................................................................................... 5
   Test Script .................................................................................................................. 5
   Component .................................................................................................................. 5
   Test Composition Environment .................................................................................. 5

2 Test Recorder ........................................................................................................... 6
   2.1 Starting the Test Creation Wizard ....................................................................... 6
   2.2 Test Creation Wizard ........................................................................................... 7
   2.3 User Authentication ............................................................................................. 8
   2.4 Test Creation Wizard - Recording ..................................................................... 10
   2.5 Checkpoints ......................................................................................................... 11

3 Cross UI Technology Scenario ............................................................................... 13
   3.1 Fiori Launchpad ................................................................................................. 13
       Adding a Checkpoint ............................................................................................... 13
       Checkpoint Input Parameters ................................................................................ 15
       Retrieving Data ...................................................................................................... 16
       Recording a New Browser Window ........................................................................ 18
       Defining Checkpoints on a Table ........................................................................... 20
       Uploading the Recorded Scenario ......................................................................... 22
       Maintaining the Test Script in TCE ....................................................................... 23
       Checking the Test Execution .................................................................................. 26

4 SAP GUI Scenarios ................................................................................................. 27
   4.1 Transactions Embedding Web Content ............................................................... 27
       Recording the Embedded HTML Content .............................................................. 27
       Tuning Checkpoint Parameters ............................................................................. 30
       Checking the Steps in TCE .................................................................................... 31
       Troubleshooting URI Issues .................................................................................. 32
       Executing the Test Script ....................................................................................... 33

5 Scenario with Asynchronous Checkpoints .............................................................. 34

6 Appendix ................................................................................................................. 36
   6.1 Checkpoints Capabilities ..................................................................................... 36
Checkpoint Operators ........................................................................................................ 36
Regular Expression Examples ............................................................................................ 37
Checkpoint Options ........................................................................................................ 38

6.2 Test Recorder Options .................................................................................................. 41
Screen Components ........................................................................................................ 41
Recording Multiple SAP GUI Sessions ............................................................................ 42
Quick Repair ..................................................................................................................... 42

7 References ....................................................................................................................... 44
7.1 Documentations .......................................................................................................... 44
7.2 SAP Notes .................................................................................................................... 44

8 Table of Figures .............................................................................................................. 45
1 Foreword

CBTA test scripts can be created by recording business scenarios. CBTA includes a test recorder that collects the events thrown by the application being tested. It generates test scripts by aggregating components.

1.1 Terminology

Test Script

A test script is an entity persisted in the repository of the Solution Manager system. The tests generated by CBTA are objects, containing:

- A list of steps to simulate user interactions
- Each step may refer to a component or another CBTA test script
- Each step may have input and output parameters.

Component

A component is the entity used to simulate user actions. Default components are those that SAP delivers. Additional components, like the screen components and view components, are generated dynamically while recording the business scenarios to be tested.

Test Composition Environment

The Test Composition Environment (TCE) is the place where CBTA test scripts are created and maintained. This UI is the one of the entry point to test automation activities. You may use it to:

- Create new test scripts
- Launch CBTA to record a scenario
- Execute existing test scripts and check the execution report
- Maintain the test script
2 Test Recorder

2.1 Starting the Test Creation Wizard

The CBTA Test Recorder can be started from the TCE. The prerequisite is to create a CBTA test script. Once created, the TCE toolbar includes a “Launch CBTA” button that lets you start the recorder.

CBTA runs client-side. Some Internet Explorer Security popups may ask for confirmation. Press the Open button to start the test recorder.
## 2.2 Test Creation Wizard

The test creation wizard is the UI shown while recording a scenario. The application being tested does not start immediately and, depending on the application type, you may have several options.

![Test Creation Wizard - Recording Options](image)

Leave the default options and press Next to start the application that is to be tested. The application starts in a new window:

- A SAP GUI session is created when testing SAP GUI transactions
- An Internet Explorer window is opened when testing web applications

### Recording Options

For details about the recording options refer the corresponding section: [Test Recorder Options](#)
2.3 User Authentication

The new session may require to authenticate the user. This is done automatically with the information specified in the test script.

The authentication differs depending on the UI technology and the system hosting the application.

- The Test Profile specified in TCE is essential here. It provides the information required to determine the user and the password (if any). Test Profiles are maintained centrally via the SUT Management UI.

- For applications hosted on an SAP System the logon is implicit. However, for some other UI technologies you may have to define a Login Schema.

Note

*For more information about Test Profiles and Login Schemas, refer to the CBTA - How-To Guide documentation.*
Figure 5: Test Script Attributes - Test Profile
2.4 Test Creation Wizard - Recording

The recording mode is activated as soon as the authentication phase is over. All actions performed against the UI of the application are recorded and the corresponding steps are shown in a list.

The toolbar lets you decide when to stop recording. You may also press pause to avoid recording some of the user interactions.

Additional options are available:

- You may add or insert a step for capturing a screenshot
- Recorded steps can be removed manually
When a step is selected, some additional information are displayed. You can, for instance, see the type of the selected control type and its URI.

Here is an example

![Figure 9: Test Creation Wizard - Step Details](image)

### 2.5 Checkpoints

The Test Engineer can define checkpoints while recording the scenario. Each checkpoint will be converted into a step in the generated test script. The component used to perform the check may vary depending on the UI technology used by the application being tested.

The test engineer can define checkpoints to verify the consistency of the application or to simply retrieve information from the UI.

- **Check Data** must be selected to define a checkpoint
- **Get Data** must be selected to retrieve information

![Figure 10: Check Picker – Check / Get Data Option](image)

**Note**

*The default components used to perform the checks are described in the documentation:*

- **CBTA Runtime Library and Default Components**
For SAP GUI transactions,

- `CBTA_GUI_GetProperty`
- `CBTA_GUI_CheckProperty`

For SAP CRM applications,

- `CBTA_CRMGetProperty`
- `CBTA_CRM_CheckProperty`
- `CBTA_CRM_GetAttribute`
- `CBTA_CRM_CheckAttribute`

For Web applications (including Web Dynpro, Web GUI, SAP UI5 / Fiori)

- `CBTA_WEB_GetProperty`
- `CBTA_WEB_CheckProperty`
- `CBTA_WEB_GetAttribute`
- `CBTA_WEB_CheckAttribute`
3 Cross UI Technology Scenario

3.1 Fiori Launchpad

While recording a scenario, the test engineer creating the test can, at any time, define a checkpoint to verify the consistency of the information being displayed.

In the example below, the Fiori App is being recorded. It starts with a launch pad where several tiles are displayed. Some of the tiles include counters, such as the Engagements and Service Delivery tile shown below. The test engineer may need to retrieve this information to check whether the value is consistent or not.

![Fiori Tile Example](image)

**Adding a Checkpoint**

You may press the “Add Checkpoint” button to start the Check Picker mode.

![Check Picker - Add Checkpoint](image)
Note

The term Check Picker is used when talking about the phase where the test engineer is about to select the IU control (or UI element).

During that phase, the test engineer must move the mouse over the application UI and wait for the control to be highlighted (with a red box) before selecting it.

Let's select the Engagements and Service Delivery tile as shown below.

The wizard creates the checkpoint and adds it to the list with some default settings. In this example, the checkpoint checks by default the title of the tile.
Checkpoint Input Parameters

The property and the operator selected by default can be modified.

Checking the title of this tile is not what we want to achieve in this example. Our goal is to retrieve the counter being displayed. Let’s change the parameters in order to retrieve the value of the counter shown by the tile.

This information is exposed via a SAP UI5 attribute of the tile that is named:

```
ui5.data.numberValue
```

Note that the value is updated when selecting another property or attribute.
You may open the list box to see all the properties or attributes that the UI control exposes.

Note

_CBT_A discovers dynamically the control properties._

_Fiori apps are built using the SAP UI5 framework. This is the reason why our tile exposes SAP UI5 attributes like shown below._

![Image of checking SAP UI5 properties](image1.png)

**Figure 17: Checking SAP UI5 Properties**

---

### Retrieving Data

In this example, we only want to retrieve the information. Let’s convert the checkpoint to a simple _Get Data_ step.

![Image of check picker with Get Data](image2.png)

**Figure 18: Check Picker - Get Data**
The corresponding step is updated and the component, which the test will used at runtime, is shown in Step Details section.

Let’s continue our scenario. We will now click the tile and start the corresponding application.
Recoding a New Browser Window

In this example, a new browser window is opened. The recording continues and actions performed in that new window are also recorded. Let’s select the second row to see how the recorder behaves.

As shown below, the Select Row step has been added to the list.

Figure 20: URI for a new Browser Window
The Step Details section provides interesting information about this new browser window. As you may notice, the URI of the step includes:

- a windowId URI attribute – this to identify the first child window of our current scenario
- The frameId URI attribute denotes the fact that the Web Dynpro content is embedded inside an IFRAME HTML element. This type of composition is quite common when using the Floorplan Manager ABAP framework. CBTA handles this situation automatically
- some URI attributes have the ls. prefix. This informs us that the new window includes a Web Dynpro application (and not a Fiori app anymore). One of the strength of CBTA is that this situation has been detected automatically and cross UI technology scenarios are supported.
- The ID of the Row Selector is quite long and includes a GUID. This GUID may vary at runtime; however, this has no impact because CBTA includes a URI resolver specialized to handle that particular use case

```
label=Row #2 - 8000035625; windowId=1; frameId=iFrameId 148772566268; ls.subtype=SC; ls.tag=TD; ls.id=SALV_WD_TABLE.ID_75CCFCE601AD96A340934A6AAA4C46C9:VIEW_TABLE.SALV_WD_UIE_TABLE.2.2
```

![FFA control](image)

**Figure 21: URI for a new Browser Window**

The new window contains a table showing the list of engagements. The purpose of our test is to verify that the number of rows displayed in the table matches the counter shown in our Engagements and Service Delivery tile.

To achieve this, we need to define a check point, selecting the table UI control. The next section explains how to proceed.
Defining Checkpoints on a Table

While defining a checkpoint, the selection of a table UI control is always a bit challenging because, most of the time, it has no border and the corresponding UI element cannot be selected.

In order to make it easier, several Check Picker modes have been introduced with CBTA 3.0 SP08.

- You may check UI relevant element
- When these modes are activated, the controls are highlighted using a different color.

Let’s click again on “Add Checkpoint” and select the table UI container of our Web Dynpro application.

When spying tables, CBTA automatically selects the rowCount property (if any).

1 Note

The row count of Web Dynpro tables is exposed via the Is.rowCount property, which is maintained by the Unified Rendering Light Speed framework.
### PFA Control

<table>
<thead>
<tr>
<th>Steps</th>
<th>Display text</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Application</td>
<td>More groups</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAP Engagement and Service Request Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Get Data</td>
<td>Engagements</td>
</tr>
<tr>
<td></td>
<td>Click</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Select row</td>
<td>Row #2 - 8000035625</td>
</tr>
</tbody>
</table>

| Check Data | Is.rowCount = 178 |

#### Step Details

- **URI**: windowId=1; frameId=iFrameId_1487772566268; ls.type=ST; tag=TABLE; ls.id=SA
- **Control type**: Sap Table
- **Default Component**: Controls\:CheckProperty

#### Check Data Details

- **Check Data**
- **Get Data**

**Parameters**

- **Property to check**: ls.rowCount
- **Operator**: =
- **Value**: 178

**Figure 23: Checkpoint - Table Row Count**
Uploading the Recorded Scenario

Our scenario is over. Let’s stop the recorder and press next to upload it to the repository (in our Solution Manager System).

![Image: Test Creation Wizard - Report]

Status: Test saved successfully.

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFA and Inspection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save Process Flow Analysis</td>
<td>Succeeded</td>
<td>Save the Process Flow Analysis in SAP Solution Manager.</td>
</tr>
<tr>
<td>Inspection</td>
<td>Succeeded</td>
<td>Inspection of screen from System Under Test.</td>
</tr>
<tr>
<td>Generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate Test</td>
<td>Succeeded</td>
<td>Generate the test from the Process Flow Analysis.</td>
</tr>
<tr>
<td>Upload</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upload of screen components</td>
<td>Succeeded</td>
<td>Save the generated screen components in SAP Solution Manager.</td>
</tr>
<tr>
<td>Upload Test</td>
<td>Succeeded</td>
<td>Save the generated test in SAP Solution Manager.</td>
</tr>
</tbody>
</table>

When back in TCE, please refresh the test script to see the updates.

Figure 24: Uploading the Recorded Scenario
Maintaining the Test Script in TCE

Once uploaded, the test script steps are generated and the corresponding test script is updated.

![Figure 25: Test Scripts Steps in TCE](image)

Our test script is correct; it contains a step per user interaction, a step to retrieve the counter from the tile and a step to check the row count of the table container.

If we run it immediately, it might run properly. However, the check does not yet verify that the row count matches the counter shown by the tile. A manual adaptation of the test script is necessary to link the two values here.

The step retrieving the tile counter relies on the CBTA_WEB_GETPROPERTY getter component. This component has a TargetField parameter that you may use to define the name of a variable, that the step will create at runtime and store in the CBTA Execution Context.

```
Note

All getter components share the same logic. Refer to the CBTA – Runtime Library and Default Components documentation for more details.
```
In this example, we will use the TargetField to create a TileCounter variable.

Switch to the Edit mode to change the TargetField parameter has shown below.

![Figure 26: Getter Components - TargetField](image)

We now need to adapt our checkpoint and to reuse the value collected by the getter component. This can be done thanks to the concept of tokens.

In this example, the token to use to get access to the value collected by the getter component is the one below:

```
%TileCounter%
```

1 Note

The token syntax is quite flexible. It also allows to evaluate expressions.
Refer to the CBTA – Runtime Library and Default Components documentation for more details.
With the changes that have been done, the test script now verifies that the application is consistent. It checks that the row count matches the counter shown within the tile and may report an error when this is not true.

The step execution creates a report with comprehensive feedback. It shows the actual value of the counter which may change, each time you run the scenario, depending on the database content.
Checking the Test Execution

Let’s have a look to the test execution report and check the overall status.

<table>
<thead>
<tr>
<th>Status</th>
<th>Step</th>
<th>Test Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>1</td>
<td>WEB_WebControl_Click</td>
<td>Operation Succeeded - Label: &quot;More groups&quot;</td>
</tr>
<tr>
<td>DONE</td>
<td>2</td>
<td>WEB_WebControl_Click</td>
<td>Operation Succeeded - Label: &quot;SAP Engagement and Service Delivery&quot;</td>
</tr>
<tr>
<td>INFO</td>
<td>3</td>
<td>StoreValue</td>
<td>%TileCounter%=172</td>
</tr>
<tr>
<td>DONE</td>
<td>4</td>
<td>WEB_WebControl_Click</td>
<td>Operation Succeeded - Label: &quot;Engagements&quot;</td>
</tr>
<tr>
<td>DONE</td>
<td>5</td>
<td>WEB_WebControl_Table_SelectRow</td>
<td>Row has been selected Operation Succeeded - Label: &quot;Row #2 - 8000035625&quot; Element Value is: &quot;8000035661&quot;</td>
</tr>
<tr>
<td>PASSED</td>
<td>6</td>
<td>WEB_WebControl_CheckProperty</td>
<td>Operation Succeeded - Element Type: &quot;TABLE&quot; Property Name: Is.rowCount Property Value: 172</td>
</tr>
<tr>
<td>INFO</td>
<td>6</td>
<td>WEB_WebControl_CheckProperty</td>
<td>Check Property - Value has been stored in the Execution Context %VIEW_TABLE.SALV.WD.UIE_TABLE.Property.Is.rowCount%=172</td>
</tr>
</tbody>
</table>

Figure 28: Execution Report

The CBTA_WEB_GetProperty component provides some feedback to let you see the collected value and the variable stored in the execution context.

Figure 29: Execution Report - StoreValue Details

The CBTA_WEB_CheckProperty component let you see the checkpoint result.

Figure 30: Execution Report - Checkpoint Details
4 SAP GUI Scenarios

4.1 Transactions Embedding Web Content

SAP GUI Transactions have the capacity to include the content of external Web applications via the \texttt{GuiHTMLViewer} control.

From a test automation perspective this kind of composition is quite challenging because the native SAP \texttt{GuiScripting} recorder does not record the actions performed against the embedded HTML content.

![GuiHTMLViewer](image)

**Figure 31:** SAP GUI Transaction - Embedding HTML Content

**Recording the Embedded HTML Content**

With CBTA 3.0 SP08, the test recorder detects automatically whether the transaction includes an \texttt{GuiHTMLViewer} container but it does not record it automatically. In such situation, a new button is shown in the test recorder toolbar.

![Test Recorder Toolbar - Attach to HTML Viewer](image)

**Figure 32:** Test Recorder Toolbar - Attach to HTML Viewer
The test engineer must explicitly click the “Attach to HTML Viewer” button and select the GuiHTMLViewer control that he/she wants to start recording.

Once selected, the corresponding step is added to the list.

Note

The component CBTA_GUI_HV_StartWebController is the one used at runtime to get access to the HTML content.

For more details, see:
- CBTA – Runtime Library and Default Components

As shown in this example, the actions performed on the HTML content are added to the list in a dedicated node.
As shown here, we have been able to define a checkpoint on the **T061** link and then click on the **Watch** button.

![Figure 35: Test Recorder - Web Content Example](image)

⚠️ Caution

*When clicking on the “Add Checkpoint” button, the Check Picker mode is not enabled automatically on the HTML content. You must select the GuiHTMLViewer control to enable it.*
**Tuning Checkpoint Parameters**

The step detail section lets you see the URI generated for the **T061** link.

In this example, this particular link has no ID and the tool determined on its own to search for it using its `innerText` property.

![Checkpoint Parameters](image)

Figure 36: Test Recorder - Checkpoint Parameters

The test engineer can then select the property to check. In this example the "exist" property has been selected to verify whether the UI element exists or not.
Checking the Steps in TCE

Back to TCE, the recorded steps can be reviewed. As shown here,

- Our script mixes SAP GUI components and WEB components.
- The CBTA_GUI_HV_StartWebController component is used to declare which GuiHTMLViewer the test execution must get access to.
- The subsequent steps are performed on the embedded HTML content.
Troubleshooting URI Issues

The Object Spy provides the ability to also spy the embedded HTML content. A new item is available in the contextual menu when a `GuiHTMLViewer` control is selected.

The Spy HTML Content menu item lets you jump to a different spy mode where the session of the embedded content is automatically selected.

Note

For more information, see the documentation:
- CBTA – Object Spy – Troubleshooting Tool

![Figure 38: How to SPY the Embedded HTML Content](image-url)
Executing the Test Script

At runtime, the CBTA_GUI_HV_StarWebController component provides some feedback in the execution report to let you check whether it could get access to the HTML content or not.

When the step succeeds, a screenshot is added to the report to let you see the actual content of the GuiHTMLViewer container.

Here is an example:

<table>
<thead>
<tr>
<th>Status</th>
<th>Step Description</th>
<th>Component</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>4 - CBTA GUI HV STARTWEBCONTROLLER</td>
<td>GetProperty</td>
<td>HtmlViewer_shell - (Handle = 3082536)</td>
</tr>
<tr>
<td>DONE</td>
<td>4 - CBTA GUI HV STARTWEBCONTROLLER</td>
<td>GS_StartWebController</td>
<td>AttachToEmbeddedContent - Operation Succeeded</td>
</tr>
<tr>
<td>INFO</td>
<td>CaptureScreen</td>
<td>Image Capture</td>
<td>&lt;no title&gt; - Captured Image</td>
</tr>
<tr>
<td>PASSED</td>
<td>5 - CBTA WEB CHECKPROPERTY</td>
<td>WEB_WebControl_CheckProperty</td>
<td>Operation Succeeded - Label: &quot;T061&quot; Property Name: exist Property Value: True</td>
</tr>
<tr>
<td>DONE</td>
<td>6 - CBTA WEB CLICK</td>
<td>WEB_WebControl_Click</td>
<td>Operation Succeeded - Label: &quot;Watch&quot;</td>
</tr>
<tr>
<td>PASSED</td>
<td></td>
<td></td>
<td>Execution Traces</td>
</tr>
</tbody>
</table>

![Figure 39: CBTA_GUI_HV_StartWebController - Execution Report](image)
5 Scenario with Asynchronous Checkpoints

Some applications may update their UI asynchronously using a job running in background. In that situation, a checkpoint can be necessary to wait for a particular state of the UI.

In the example below, the application starts in a state where all buttons are disabled because the retrieval of the data takes some time. The “Workload Analysis” button gets enabled only when the page rendering is complete.

![Figure 40: Asynchronous Checkpoint Example](image)

The end-user needs to wait for the page to be complete before doing any actions.

From a test automation perspective this is challenging because the recorder is not aware of this background job. That is to say that the recorded script may fail because it tries to click on the “Workload Analysis” button before it gets enabled. This problem can be solved by changing the options to define an asynchronous checkpoint.

![Figure 41: Asynchronous Checkpoint Definition](image)
As shown here, the checkpoint will check for the Light Speed property providing the state of the button. The check will be done at regular interval until a timeout is reached.

If the button state matches the expected state before the timeout the execution flow will resume and the test execution may succeed. It the state does not match after the time out, the execution flow is interrupted and the execution report will show the error.

These options can be changed while recording as well. Check the “Data Checked Asynchronously” checkbox to enable them.

You may check the parameters of the step which is generated when uploading the test script to the solution manager system.

The component parameters (as shown in TCE) are the ones below:

- the PropertyName parameter is set ls.data.enabled
- The Options parameter includes the specific options of the CBTA_WEB_CheckProperty component to specify the initial wait time, the interval and the time out.
6 Appendix

6.1 Checkpoints Capabilities

Checkpoint Operators

The operators, which the CheckProperty and CheckAttribute components support, are listed below.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals</td>
</tr>
<tr>
<td>&lt;</td>
<td>Lower than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
<tr>
<td>{contains}</td>
<td>Contains</td>
</tr>
<tr>
<td>{startsWith}</td>
<td>Starts with</td>
</tr>
<tr>
<td>{endsWith}</td>
<td>Ends with</td>
</tr>
<tr>
<td>{matches}</td>
<td>Matches a regular expression</td>
</tr>
</tbody>
</table>

Negative Operators (new 3.0.8):

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{!contains}</td>
<td>Does not contain</td>
</tr>
<tr>
<td>{!startsWith}</td>
<td>Does not start with</td>
</tr>
<tr>
<td>{!endsWith}</td>
<td>Does not end with</td>
</tr>
<tr>
<td>{!matches}</td>
<td>Does not match a regular expression</td>
</tr>
</tbody>
</table>

While recording, a list box shows the list of the Boolean operators that you may use.
Regular Expression Examples

When using the `{matches}` operator the value must be specified using the .NET regular expression syntax.

**Example**

Regular expression:

```
^Sales order [0-9]+ has been created$
```

Matches values of the following form:

Sales order 123456 has been created

But, it does not match the one below:

Sales order has been created

**Example**

You may use a regular expression to check the `ui5.data.numberUnit` property exposed by a SAP UI5 tile, like in this example:

![Figure 45: Unit Property of a SAP UI5 tile](image1)

![Figure 46: Check using a Regular Expression](image2)
Checkpoint Options

The options supported when checking UI element properties are common to all UI technologies (including SAP GUI).

They are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/u (for uppercase)</td>
<td>Both values are converted to upper-case before being compared</td>
</tr>
<tr>
<td>/t (for trimmed)</td>
<td>Both values are trimmed before being compared</td>
</tr>
<tr>
<td>/i (integer)</td>
<td>Both values are converted to an integer before being compared</td>
</tr>
<tr>
<td>/f (float)</td>
<td>Both values are converted to a float (double) before being compared</td>
</tr>
<tr>
<td>/b (bool)</td>
<td>Both values are converted to a Boolean before being compared</td>
</tr>
</tbody>
</table>

Language-Dependent Comparison and Conversion Issues

The locale of the VB script interpreter depends on the language settings of the operating system.

This may have an effect when converting string values to numerical or date values. To address conversion issues, make sure the regional settings of the Operating System and the SAP GUI settings are the same.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/d (date)</td>
<td>Both values are converted to a date before being compare (new 3.0.8)</td>
</tr>
<tr>
<td>/CC</td>
<td>Or /CustomConversion (new 3.0.8) The conversion is performed by calling the function GS_CustomConvert.</td>
</tr>
</tbody>
</table>

The default implementation does nothing. The test engineer must provide its own implementation by overriding the default implementation that SAP delivers. This must be done via the Runtime Library Manager. The function is called for each operand. It receives two input parameters:

- The actual operand value (before conversion)
- The options specified – (all options including the /CC)

The checkpoint options are shown in the “Step Details” section.

- Some options can be set by default according to the type of the selected property.
- For instance, /b is set by default when checking the existence of the UI control
- You may press “Options” button to change the default values
This opens a popup, where all options can be set manually.
Asynchronous Checkpoint Options

The options supported when checking UI element properties and attributes are common to all Web UI technologies.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wsd(?s)</td>
<td>Time to wait before doing the first check (expressed in seconds) Example /wsd2s (for two seconds)</td>
</tr>
<tr>
<td>/wi(?s)</td>
<td>Wait Interval - Time to wait between two checks</td>
</tr>
<tr>
<td>/wx(?s)</td>
<td>Max wait time – Defines the time out. Execution flow is interrupted when the expected state is not met after the timeout.</td>
</tr>
</tbody>
</table>

These options can be changed while recording as well. Check the “Data Checked Asynchronously” checkbox to enable them.

The screenshot below shows the input parameters of a checkpoint waiting for a Web Dynpro Button to be enabled.

![Figure 49: Check Picker - Asynchronous Checkpoint Options](image)

The screenshot below shows the input parameters of a checkpoint waiting for a Web Dynpro Button to be enabled.

![Figure 50: Input Parameters for an Asynchronous Checkpoint](image)
6.2 Test Recorder Options

Screen Components

CBTA delivers a set of components, which are used to simulate user interactions when executing a test script. These components perform atomic operations such as entering values, marking a checkbox, selecting an item in a dropdown list box. When populating SAP GUI screens, the number of interaction can be huge and it may make sense to group them in a single step; and this is the purpose of generating Screen Components.

![Figure 51: Test Recorder Wizard - Use Screen Components](image)

Mark the checkbox to enable the Inspection process. This process is responsible for collecting the list of input fields that the screen includes. Based on this information a Screen Component is generated and uploaded to the test repository.

*Note*

The inspection process may generate a screen component per Dynpro Screen when testing SAP GUI Transactions. For CRM Web applications and Web Dynpro ABAP, the term View Component is used instead. View Components are generated as soon as a form is populated.
Recording Multiple SAP GUI Sessions

When testing SAP GUI transactions, the test recorder only records actions performed on the main SAP GUI window. In other words, new SAP GUI sessions started from the main window are not recorded by default.

You may enable the recording of all SAP GUI sessions by checking the checkbox shown below.

![Screen Component Options](image)

The URI generated for the steps targeting a different session may include a `sessionId` URI attribute.

Here is an example:

```
label=Exit (Shift+F3); sessionId=1; type=GuiButton; id=wnd[0]/tbar[0]/btn[15]
```

**Note**

*The same option must be selected when recording scenarios jumping from a web application to a SAP GUI transaction.*

Quick Repair

One of the test automation challenges is that the existing test scripts have, from time to time, to be maintained. This typically happens when a new version of the application being tested is deployed. In such situation, one of the option is to use the *Object Spy* and troubleshoot the test execution issue. Unfortunately, this approach can be very difficult when the test scripts have hundreds of steps.

With CBTA 3.0 SP05, you may start the recorder in different modes instead. These mode allows you to maintain the existing test scripts by re-recording only a subset of the scenario. The newly created steps can be added at the end of the existing test script or inserted after a particular step.
Recording Options

![Recording Options](image)

Figure 53: Test Recorder - Quick Repair Options

Note that these options are disabled by default. The prerequisites to have them enabled are the following:

- You must first execute an existing test script
- The window opened by the test script execution must be preserved - do not close them manually!
- The test execution must be successful – the overall status of the report must be PASSED or DONE.
- You must then start the test recorder from the same test script – i.e.: the one used for the execution

When all these pre-requisites are fulfilled the recording options are available.

⚠️ Caution

*In order to be able to insert the steps at a particular position you must first select that step and run the test script by selecting the “Execute to Step” option like shown below.*

![Test Script Steps](image)

Figure 54: TCE Option - Execute to Step
7 References

7.1 Documentations

Some additional documents have to be considered to take benefit of the CBTA capabilities.

<table>
<thead>
<tr>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBTA – User Guide</td>
</tr>
<tr>
<td>CBTA – Runtime Library and Default Components</td>
</tr>
<tr>
<td>CBTA – Object Spy – Troubleshooting Tool</td>
</tr>
</tbody>
</table>

7.2 SAP Notes

The following table list the SAP Notes mentioned in this document.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2133396</td>
<td>CBTA - Test Automation Tool - Object Spy - Session not compliant with test automation requirements</td>
</tr>
<tr>
<td>1666201</td>
<td>WebCUIF - Collective Note to enable the Test Mode for CRM Web UIs</td>
</tr>
<tr>
<td>2177107</td>
<td>Test Automation of Web Dynpro scenarios - Stable ID Mode not propagated to child windows</td>
</tr>
</tbody>
</table>
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Launch CBTA from TCE</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Confirm Security Popups</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Test Creation Wizard - Recording Options</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>User Authentication Phase</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Test Script Attributes - Test Profile</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Test Creation Wizard - Recorded Steps</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Test Recorder Toolbar</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Test Recorder - Contextual Menu</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>Test Creation Wizard - Step Details</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>Check Picker - Check / Get Data Option</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>Fiori Tile Example</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>Check Picker - Add Checkpoint</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>Check Picker Selection</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>Checkpoint Example</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Checkpoint Parameters</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Checkpoint Details</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>Checking SAP UIS Properties</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>Check Picker - Get Data</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>Check Picker - Get Data Step</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>URI for a new Browser Window</td>
<td>18</td>
</tr>
<tr>
<td>21</td>
<td>URI for a new Browser Window</td>
<td>19</td>
</tr>
<tr>
<td>22</td>
<td>Check Picker - TABLE Mode</td>
<td>20</td>
</tr>
<tr>
<td>23</td>
<td>Checkpoint - Table Row Count</td>
<td>21</td>
</tr>
<tr>
<td>24</td>
<td>Uploading the Recorded Scenario</td>
<td>22</td>
</tr>
<tr>
<td>25</td>
<td>Test Scripts Steps in TCE</td>
<td>23</td>
</tr>
<tr>
<td>26</td>
<td>Getter Components - TargetField</td>
<td>24</td>
</tr>
<tr>
<td>27</td>
<td>Input Parameter using a Token</td>
<td>25</td>
</tr>
<tr>
<td>28</td>
<td>Execution Report</td>
<td>26</td>
</tr>
<tr>
<td>29</td>
<td>Execution Report - StoreValue Details</td>
<td>26</td>
</tr>
<tr>
<td>30</td>
<td>Execution Report - Checkpoint Details</td>
<td>26</td>
</tr>
<tr>
<td>31</td>
<td>SAP GUI Transaction - Embedding HTML Content</td>
<td>27</td>
</tr>
<tr>
<td>32</td>
<td>Test Recorder Toolbar - Attach to HTML Viewer</td>
<td>27</td>
</tr>
<tr>
<td>33</td>
<td>Test Recorder - Start Web Controller</td>
<td>28</td>
</tr>
<tr>
<td>34</td>
<td>Test Recorder - Web Application Steps</td>
<td>28</td>
</tr>
<tr>
<td>35</td>
<td>Test Recorder - Web Content Example</td>
<td>29</td>
</tr>
<tr>
<td>36</td>
<td>Test Recorder - Checkpoint Parameters</td>
<td>30</td>
</tr>
<tr>
<td>37</td>
<td>TCE - CBTA_GUI_HV_StartWebController</td>
<td>31</td>
</tr>
<tr>
<td>38</td>
<td>How to SPY the Embedded HTML Content</td>
<td>32</td>
</tr>
<tr>
<td>39</td>
<td>CBTA_GUI_HV_StartWebController - Execution Report</td>
<td>33</td>
</tr>
<tr>
<td>40</td>
<td>Asynchronous Checkpoint Example</td>
<td>34</td>
</tr>
<tr>
<td>41</td>
<td>Asynchronous Checkpoint Definition</td>
<td>34</td>
</tr>
<tr>
<td>42</td>
<td>Check Picker - Initial Wait Time and Timeout Definition</td>
<td>35</td>
</tr>
<tr>
<td>43</td>
<td>TCE - Input Parameters for an Asynchronous Checkpoint</td>
<td>35</td>
</tr>
<tr>
<td>44</td>
<td>Checkpoint Operators</td>
<td>37</td>
</tr>
<tr>
<td>45</td>
<td>Unit Property of a SAP UIS tile</td>
<td>37</td>
</tr>
<tr>
<td>46</td>
<td>Check using a Regular Expression</td>
<td>37</td>
</tr>
<tr>
<td>47</td>
<td>Check Picker - Changing Options</td>
<td>39</td>
</tr>
<tr>
<td>48</td>
<td>Check Picker - Options</td>
<td>39</td>
</tr>
</tbody>
</table>
Figure 49: Check Picker - Asynchronous Checkpoint Options .......................................................... 40
Figure 50: Input Parameters for an Asynchronous Checkpoint ......................................................... 40
Figure 51: Test Recorder Wizard - Use Screen Components ............................................................. 41
Figure 52: Recording Multiple SAP GUI Sessions ............................................................................ 41
Figure 53: Test Recorder - Quick Repair Options ............................................................................ 43
Figure 54: TCE Option - Execute to Step ....................................................................................... 43