# Table of Contents

## 1 Foreword

## 2 Terminology

2.1 Test Script

2.2 Component

2.3 Test Composition Environment

2.4 Test Recorder

2.5 Test Player

2.6 Object Spy

## 3 Runtime Library Concepts

3.1 URI - Uniform Resource Identifier

3.2 CBTA Default Components

3.3 Execution Context

3.4 Tokens and Token Resolution

## 4 Keywords

4.1 Keywords for Checking Conditions

Keyword: IF

Keyword: ELSE

Keyword: ELSE
4.2 DO / LOOP Iteration Keywords ................................................................. 39
Keyword: DO ........................................................................................................ 40
Keyword: LOOP ...................................................................................................... 40
Keyword: EXIT_DO ............................................................................................... 40

4.3 FOR / NEXT Iteration Keywords ................................................................. 41
Keyword: FOR ........................................................................................................ 42
Keyword: NEXT ...................................................................................................... 42
Keyword: EXIT_FOR ............................................................................................ 42

5 Components ..................................................................................................... 44
5.1 Components Common to All UI Technologies ............................................. 44
CBTA_A_GetFromExecutionCtxt ......................................................................... 44
CBTA_A_SetInExecutionCtxt .............................................................................. 45
CBTA_GUI_A_ReportMessage ........................................................................... 46
CBTA_A_CompareValues ..................................................................................... 47
CBTA_A_SetCondition (deprecated) ................................................................. 48
CBTA_A_RemoveCondition (deprecated) ............................................................ 49
CBTA_A_Wait ....................................................................................................... 49

6 SAP GUI Transactions ...................................................................................... 50
6.1 Identifying SAP GUI Controls ..................................................................... 50
6.2 SAP GUI - Action Components ................................................................... 52
CBTA_GUI_A_CaptureScreen ............................................................................ 52
CBTA_GUI_A_CheckTCode ............................................................................... 52
CBTA_GUI_A_CloseWindow .............................................................................. 52
CBTA_GUI_A_EndTransaction ............................................................................ 53
CBTA_GUI_A_ExecuteStatement ....................................................................... 54
CBTA_GUI_A_Invoke_Function .......................................................................... 55
CBTA_GUI_A_LaunchAndLogin ........................................................................ 56
CBTA_GUI_A_LogOff ......................................................................................... 56
CBTA_GUI_A_PressKey ..................................................................................... 56
CBTA_GUI_A_StartTransaction ........................................................................ 57

6.3 SAP GUI - Generic Components for SAP GUI ........................................... 58
CBTA_GUI_CheckProperty ............................................................................... 58
CBTA_GUIGetProperty ....................................................................................... 59
CBTA_GUI_GetText ............................................................................................ 60
CBTA_GUI_SelectContextMenuItem .................................................................... 61
CBTA_GUI_SetFocus
CBTA_GUI_SetProperty
CBTA_GUI_SetText

6.4 SAP GUI - Control Components for SAP GUI

CBTA_GUI_BTN_PressButton
CBTA_GUI_CB_GetSelected
CBTA_GUI_CB_SetSelected
CBTA_GUI_CB.GetKey
CBTA_GUI_CB.GetValue
CBTA_GUI_CB_SetKey
CBTA_GUI_CB_SetValue
CBTA_GUI_GV_ClearSelection
CBTA_GUI_GV_ClickCurrentCell
CBTA_GUI_GV_DeleteRows
CBTA_GUI_GV_DeselectColumn
CBTA_GUI_GV_DoubleClickCell
CBTA_GUI_GV_DuplicateRows
CBTA_GUI_GV_FindRow
CBTA_GUI_GV_GetCellChecked
CBTA_GUI_GV_GetCellState
CBTA_GUI_GV_GetCellValue
CBTA_GUI_GV_ModifyCell
CBTA_GUI_GV_ModifyCheckBox
CBTA_GUI_GV_MoveRows
CBTA_GUI_GV_PressButton
CBTA_GUI_GV_PressColumnHeader
CBTA_GUI_GV_Press_Enter
CBTA_GUI_GV_Press_F1
CBTA_GUI_GV_Press_F4
CBTA_GUI_GV_TB_PressButton
CBTA_GUI_GV_TB_PressMenuItem
CBTA_GUI_GV_SelectAll
CBTA_GUI_GV_SelectColumn
CBTA_GUI_GV_SelectMenuItem
CBTA_GUI_GV_SetCurrentCell
CBTA_GUI_GV_SetSelectedRows
CBTA_GUI_M_Select
CBTA_GUI_HV_StartWebController
CBTA_GUI_PF_SetSecureText........................................................................85
CBTA_GUI_RB_GetSelected.........................................................................86
CBTA_GUI_RB_SelectRadioButton.............................................................87
CBTA_GUI_RB_SetSelected ......................................................................87
CBTA_GUI_SB_GetMessageParam..............................................................88
CBTA_GUI_SB_GetMessageParams.............................................................88
CBTA_GUI_SB_GetMessageType..................................................................90
CBTA_GUI_T_SelectTab..............................................................................91
CBTA_GUI_TS_GetSelectedTab...................................................................91
CBTA_GUI_TC_GetCellData........................................................................92
CBTA_GUI_TC_SetCellData........................................................................93
CBTA_GUI_TC_IsRowSelected....................................................................94
CBTA_GUI_TC_FindRow............................................................................95
CBTA_GUI_TC_SelectRow...........................................................................96
CBTA_GUI_TXTE_DoubleClick...................................................................96
CBTA_GUI_TXTE_Press_F4.........................................................................96
CBTA_GUI_TB_PressButton.......................................................................97
CBTA_GUI_TB_PressCtxtButton.................................................................97
CBTA_GUI_TB_SelectMenuItem.................................................................98
CBTA_GUI_T_ChangeCheckbox..................................................................99
CBTA_GUI_T_ClickLink............................................................................100
CBTA_GUI_T_CollapseNode.....................................................................101
CBTA_GUI_T_DoubleClickItem..................................................................101
CBTA_GUI_T_DoubleClickNode..................................................................101
CBTA_GUI_T_ExpandNode..........................................................................102
CBTA_GUI_T_GetCheckBoxState...............................................................103
CBTA_GUI_T_PressButton.........................................................................104
CBTA_GUI_T_PressedHeader....................................................................104
CBTA_GUI_T_SelectColumn.......................................................................105
CBTA_GUI_T_SelectColMenuItem..............................................................106
CBTA_GUI_T_SelectMenuItem...................................................................108
CBTA_GUI_T_SelectItem............................................................................109
CBTA_GUI_T_SelectNode..........................................................................110
CBTA_GUI_T_SetCheckBoxState...............................................................110
CBTA_GUI_T_UnselectAll..........................................................................111
CBTA_GUI_T_UnselectColumn...................................................................111
CBTA_GUI_T_UnselectNode.......................................................................112

6.5 SAP GUI - Test Automation Challenges .............................................113
7 SAP CRM / WebCUIF ................................................................. 126

7.1 URI Identifying CRM UI Elements ........................................ 126

7.2 SAP CRM - Action Components .......................................... 128

CBTA_CRM_A_CaptureScreen ................................................ 128
CBTA_CRM_A_GetLastMsgParams ........................................... 128
CBTA_CRM_A_GetMessageParams .......................................... 129
CBTA_CRM_A_LaunchAndLogin .............................................. 130
CBTA_CRM_A_LogOff .......................................................... 131
CBTA_CRM_A_ClosePopup .................................................... 132

7.3 SAP CRM - Generic Components ....................................... 133

CBTA_CRM_CheckAttribute .................................................. 133
CBTA_CRM_CheckProperty ................................................... 134
CBTA_CRM_Click .............................................................. 135
CBTA_CRM_GetAttribute ..................................................... 135
CBTA_CRM_GetProperty ..................................................... 136
CBTA_CRM_PressKey .......................................................... 137
CBTA_CRM_SetAttribute ..................................................... 137
CBTA_CRM_SetProperty ..................................................... 137
CBTA_CRM_SetFocus .......................................................... 138

7.4 SAP CRM - Control Components ....................................... 139

CBTA_CRM_BTN_ClickButton ............................................... 139
CBTA_CRM_BTN_SetButtonState ........................................... 139
CBTA_CRM_CB_GetSelected ................................................ 140
CBTA_CRM_CB_SetSelected ................................................ 140
CBTA_CRM_DP_OpenDatePicker ........................................... 140
CBTA_CRM_DP_SelectDate ................................................... 141
CBTA_CRM_DLB_SelectItem ................................................ 141
CBTA_CRM_DLB_SelectKey .................................................. 141
CBTA_CRM_DLB_SelectValue .............................................. 141
CBTA_CRM_IF_GetValue ..................................................... 142
CBTA_CRM_IF_OpenInputHelp ............................................. 142
CBTA_CRM_IF_SetValue ..................................................... 142
8 Web Applications .................................................................................. 151
  8.1 URI Identifying HTML UI Elements ............................................. 152
      HTML Elements and Documents ............................................. 152
      Web Controls versus HTML Elements ................................ 152
      Web Dynpro Controls ......................................................... 153
      Web GUI Controls .................................................................. 154
      Java Web Dynpro Controls ................................................... 154
      WebCUIF Controls (for SAP CRM Web Applications) .......... 155
      SAP UIS and FIORI Controls ................................................. 155
  8.2 URI Resolution Strategies .............................................................. 157
      URI Resolution Ambiguities .................................................... 158
      Searching using Regular Expressions ................................... 159
  8.3 Web UI Technology – Action Components .................................. 162
      CBTA_WEB_A_CaptureScreen ............................................. 162
      CBTA_WEB_A_GetMessageParams ..................................... 163
      CBTA_WEB_A_ExecuteStatement ....................................... 164
      CBTA_WEB_A_Invoke_Version ............................................ 166
      CBTA_WEB_A_CloseWindow ............................................... 167
      CBTA_WEB_A_LogOff .......................................................... 167
  8.4 Web UI Technology – Generic Components ................................ 168
      CBTA_WEB_CRM.SelectTransactionType ................................ 168
      CBTA_WEB_CRM.SelectMenuItemByText ............................... 169
      CBTA_WEB_CRM.SelectRadioButton ................................. 170
      CBTA_WEB_CRM.SelectMenuItem ....................................... 170
      CBTA_WEB_CRM.SelectAttribute ........................................ 171
### CBTA Class

Function GetSAPGUIConnection() ......................................................... 193
Function GetSAPGUISession() ........................................................... 194
Function GetControl( URI ) ............................................................. 194
Function ResolveParameterValue( Value ) ......................................... 194
Sub Report( Severity, Topic, Message, Options ) ............................ 194

### 8.5 Web UI - Test Automation Challenges

Handling of Internet Explorer Windows ............................................. 175
Internet Explorer Security Popups ...................................................... 176

### 9 Web Dynpro / Light Speed .......................................................... 178

#### 9.1 Light Speed - Action Components .......................................... 178

CBTA_LSA_GetMessageParams ........................................................ 178

#### 9.2 Light Speed - Control Components ......................................... 179

CBTA_LS_T_FindRow ........................................................................ 179
CBTA_LS_T_SetFilterValue ............................................................. 184
CBTA_LS_T_SetFilterValues ........................................................... 185
CBTA_LS_T_SetCellValue ............................................................... 186
CBTA_LS_T_SetCellValues ............................................................. 187

### 10 SAP UI5 / FIORI ........................................................................ 188

#### 10.1 SAP UI5 - Action Components ............................................ 188

CBTA_UI5_A_GetMessage (new 3.0.8) ............................................. 188
CBTA_UI5_A_GetMessageParams (new 3.0.8) .................................. 190

#### 10.2 SAP UI5 - Control Components ............................................ 191

CBTA_UI5_T_FindRow (new 3.0.8) ................................................... 191

### 11 Runtime Library API .................................................................. 193

#### 11.1 CBTA Class

Function GetSAPGUIConnection() ......................................................... 193
Function GetSAPGUISession() ........................................................... 194
Function GetControl( URI ) ............................................................. 194
Function ResolveParameterValue( Value ) ......................................... 194
Sub Report( Severity, Topic, Message, Options ) ............................ 194
Sub Log(message) ........................................................................................................ 195
Sub CaptureScreen() .................................................................................................. 195
Sub Wait(milliseconds) ............................................................................................... 195
Sub LoadLibrary(Library) ............................................................................................. 195

12 References ............................................................................................................. 197
12.1 Documentations .................................................................................................. 197
12.2 SAP Notes .......................................................................................................... 197

13 Table of Figures .................................................................................................... 198
Component-Based Test Automation (CBTA) includes a runtime library which is used while executing a test script.

The runtime library consists of:

- Libraries – a set of VB script files providing the core test execution features
- Default components – components that simulate user actions when testing business applications.

The runtime library supports several UI technologies:

- **SAP GUI** – used by SAP R/3 applications
- **WebCUIF** – used by SAP CRM (CRM web applications)
- **Web** – displays content using HTML tags like:
  - **BSP**
- **Unified Rendering Light Speed (LS)** – UI layer common to most SAP UI frameworks, like:
  - **Web Dynpro ABAP**
  - **Web Dynpro Java** - (version based on Light Speed)
  - **Web GUI** – (a.k.a. SAPGUI for HTML)
- Applications based on SAP UI5 (including FIORI applications)

Note

The runtime library and the default components are part of the ST software component.

To benefit from the latest improvements, you may implement the following SAP Note:

- SAP Note [2029868](#) - CBTA - Runtime Library - Fixes & Improvements
2 Terminology

2.1 Test Script

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

- A list of steps to simulate user interactions
- Each step refers to a component
- Each step may have input and output parameters.

2.2 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.

A component contains VB script coding to call the component implementation that the CBTA runtime library provides.

Note that IF, ELSE and ENDIF are also delivered as default components even though they should be considered as keywords.

2.3 Test Composition Environment

The Test Composition Environment (TCE) is the place where CBTA test scripts are created and maintained.
Figure 1: Test Repository Tile
The TCE UI is the main entry point to test automation activities.

You may use it to:

- Create new test scripts
- Start the CBTA Test Recorder
- Execute existing test scripts and check the execution report
- Maintain the test script steps and tune the corresponding input parameters

### 2.4 Test Recorder

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.

The test recorder can be started from the TCE UI and allows you to define checkpoints.

---

**Documentation**

*For more details, refer to the documentation:

CBTA - Test Recorder*

---

![CBTA Test Configuration: ZPP_URL_ITSM_MYIN](image)

---

**Test Script Steps**

- CBTA Default Component: CBTA_WEB_CLICK

---

**Figure 2: Launch CBTA from TCE**

### 2.5 Test Player

When executing a CBTA test script, the corresponding VBScript coding is built by aggregating the content of each step and sent to the client computer. A VBScript interpreter is used client-side to execute the script.

- The test player relies on a built-in wait mechanism to automatically wait for the application to be ready before performing the next actions.
The test player discovers dynamically the UI technology used by the application. You may start a scenario from a Fiori app and jump from it to a Web Dynpro or SAP UI5 application; in the same or in a new window.

Components are specialized per UI technology in order to benefit from the specific properties and attributes that the UI framework may expose.

### 2.6 Object Spy

The Object Spy is a tool that CBTA delivers to troubleshoot test execution issues.

It can be launched with the button “Get Technical UI Information”, which is visible when a component is selected in the Test Script tab of TCE.

![Test Composition Environment](image)

**CBTA Test Configuration: ZPP_URL_ITSM_MYINCIDENT2 - ZPP_URL_ITSM_MYINCIDENT2**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Test Script</th>
<th>Parameters</th>
<th>Test Data</th>
<th>Search Terms</th>
<th>Administrative Data</th>
</tr>
</thead>
</table>

**CBTA Test Configuration Steps**

<table>
<thead>
<tr>
<th>Test Script Steps</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>View: [Standard View]</td>
<td>Print Version</td>
<td>Up</td>
<td>Down</td>
<td>Move</td>
<td>Copy</td>
</tr>
<tr>
<td>CBTA Default Component</td>
<td>CBTA_WEB_CLICK</td>
<td>[Click on &quot;My Incidents&quot;] - Control type is sap.m.GenericTile</td>
<td>[Click on &quot;Add&quot;] - Control type is sap.m.Button</td>
<td>[Click on &quot;2MIN&quot;] - Control type is sap.m.Text</td>
<td>[Click on &quot;Create&quot;] - Control type is sap.m.Button</td>
</tr>
</tbody>
</table>

**Details of CBTA Default Component CBTA_WEB_CLICK**

<table>
<thead>
<tr>
<th>Test Tool</th>
<th>CBTA Version</th>
<th>Priority</th>
<th>Medium</th>
<th>Active</th>
<th>Application Component</th>
<th>3V-SIMO-TWB</th>
<th>Responsible</th>
<th>ROSS1CL</th>
</tr>
</thead>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>View: [Standard View]</th>
<th>Print Version</th>
<th>Set Usage</th>
<th>Get Technical UI Information</th>
<th>URI</th>
<th>Mass update</th>
</tr>
</thead>
</table>

**Documentation**

The object spy capabilities are documented in the document:

CBTA – Object Spy – Troubleshooting Tool

![Figure 3: Starting the Object Spy from TCE](image)
3 Runtime Library Concepts

3.1 URI - Uniform Resource Identifier

CBTA tests applications by performing actions against their UI (user interface). One prerequisite of this approach is to be able to uniquely identify controls within the hierarchy of the UI elements being displayed. The information required to identify a control unambiguously may vary depending on the underlying UI technology. The runtime library uses the concept of uniform resource identifier (URI).

The URI syntax is well defined and quite flexible to cover complex scenarios including multiple windows and frames. The information required to search for the UI elements is specialized per IU technology.

- For SAP GUI applications see section Identifying SAP GUI Controls
- For CRM applications see section Identifying CRM UI Elements
- For Web, Web Dynpro ABAP, Web Dynpro Java, Web GUI, SAP UI5 and FI/IR refer to section Identifying HTML UI Elements

Note that a default URI is determined automatically when recording the scenario. When the URI generated by default is not reliable, another URI might be necessary. URI alternatives can be determined using the Object Spy.

There are situations where the Runtime Library that SAP delivers is not sufficient. Complex scenarios may require that the test engineers write some custom functions. This is possible by customizing the Runtime Library via the Runtime Library Manager.

Documentation

Additional technical documentations are explaining how to customize the runtime library. Please refer to:

- CBTA – Runtime Library Manager – CBASE Customization
- CBTA – Custom Code Patterns – CBASE Customization
- CBTA – Test Automation - Query API

URI Syntax

A URI consists of:

- One or more URI fragments
- The separator to use between fragments is: " > " (with a leading and trailing space character)

Each URI fragment consists of:

- One or more URI attributes

Each URI Attribute consists of:

- A name/value pair
- The separator to use between two pairs is: "; " (with a trailing space character)
Well-formatted URI Examples:

URI with a single fragment:

\[
\text{type=BUTTON; name=SAVE}
\]

URI with two fragments:

\[
id=\text{WORKAREA\_FRAME}; \text{type=IFRAME} \rightarrow \text{type=BUTTON; id=BUTTON\_SAVE}
\]

URI Resolution

The URI is resolved at runtime, while executing the steps of a test script. The test player parses the URI, and for each URI fragment, searches the application content for the corresponding UI control (or UI element).

\[
\begin{align*}
\text{Note} & \\
\text{In general, the term UI control is used when targeting something that the end user can see – such as a button, a checkbox, etc.} \\
\text{The term UI element refers to something which is not necessarily visible or something the end user is not aware of. UI controls can be an aggregation of several UI elements.}
\end{align*}
\]

Each fragment identifies a UI element:

- The first fragments identify UI containers (if any)
- The last fragment identifies the target – the UI element the test wants to interact with.

For SAP GUI Transactions, only one fragment is necessary. The URI syntax is then quite simple. However, for Web applications, the generated content can be very sophisticated. The HTML language allows page composition using FRAMES and IFRAMES, so a single fragment is not always sufficient.

Here is an example of page composition in which several frames are embedded in the main document that the application displays.

![Figure 4: Page Composition Example](image-url)
The URI to target the input field would have the following form:

Short form:

- label=Order Type; frameld=Frame2; id=Field1; tag=INPUT

Full syntax using fragments:

- id=Frame1; tag=FRAME > id=Frame2; tag=IFRAME > label=Order Type; id=Field1; tag=INPUT

The two examples shown here are equivalent. The difference between the two is that the second syntax is stricter than the first one, since Frame2 will only be found when it is a child frame of Frame1. With the short syntax, Frame2 will be found whatever its container is. This is convenient but may lead to unpredictable results with scenarios in which Frame2 is embedded twice.

**URI Attributes**

As already mentioned, the URI syntax is quite flexible. This is the reason why we cannot provide an exhaustive list of URI attributes. The test player discovers at runtime the attributes exposed by the targeted UI control (or UI element). This list of attributes is therefore dynamic and differs depending on the UI control nature.

**Wait and Attempts URI Attributes**

The CBTA test player relies on a built-in wait mechanism to automatically wait for the application to be ready before performing the next actions. This implicit wait mechanism only works for SAP UI technologies and you may face situations where an additional wait time must be defined. This can be done using two specific URI attributes defining how long to wait and retry when searching for a UI control:

- *wait* – defines the time to wait in milliseconds
- *attempts* – defines the number of attempts to perform when searching for the UI control

The wait time is used before the first attempt but also between two attempts.

- The test player performs the action as soon as the UI control is found.
- The test player gives up when the UI control is not found after having performed all attempts.

**Example:**

With this example, the test script may try 10 times to search for the control and may give up after ten seconds.

- **wait=1000; attempts=10; id=Frame1; tag=FRAME > label=Order Type; id=Field1; tag=INPUT**

Note that these attributes can be specified in the first fragment only.
### 3.2 CBTA Default Components

CBTA relies on the concept of components. Most of the components are used to simulate user interactions. Some others are used to verify the application consistency; they get information from the UI and provide the ability to perform checkpoints.

CBTA test scripts are not built by writing any coding. They are built by aggregating components. Each component instance is then a step of the test script. At runtime, an error is reported in the execution report as soon as one of the steps fails.

**Component Types**

We distinguish several types of components.

<table>
<thead>
<tr>
<th>Component Type</th>
<th>Description</th>
<th>Naming (depending on the UI Technology)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>Components of this type do not target a UI control.</td>
<td>CBTA_&lt;UITech&gt;<em>A</em>&lt;Action&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_GUI_A_LogOff</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_WEB_A_LogOff</td>
<td></td>
</tr>
<tr>
<td>GENERIC</td>
<td>Components of this type have (at least) a URI parameter which is used to identify the target.</td>
<td>CBTA_&lt;UITech&gt;_&lt;Action&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_WEB_SetValue</td>
<td></td>
</tr>
<tr>
<td>GETTER</td>
<td>Components of this type are used to retrieve information from a UI control or element.</td>
<td>CBTA_&lt;UITech&gt;<em>&lt;ControlType&gt;</em>&lt;Action&gt;</td>
</tr>
<tr>
<td></td>
<td>• They have one or more output parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• They also store the retrieved information in the execution context.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Some of them can also check whether the actual value is correct. They report an error when the value does not match the expected one.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Getters:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_&lt;UITech&gt;_GetAttribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_&lt;UITech&gt;_GetProperty</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checkpoints:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_&lt;UITech&gt;_CheckAttribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_&lt;UITech&gt;_CheckProperty</td>
<td></td>
</tr>
<tr>
<td>CONTROL</td>
<td>Components of this type are specialized for a particular control type.</td>
<td>CBTA_&lt;UITech&gt;_&lt;ControlType&gt;<em>A</em>&lt;Action&gt;</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_GUI_GV_FindRow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_CRM_T_FindRow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_LS_T_FindRow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CBTA_UI5_T_FindRow</td>
<td></td>
</tr>
</tbody>
</table>
Standard Component Behavior

The runtime library tries to apply the same logic to all components.

For example:

- The same exception handling provides comprehensive feedback to the user when an exception occurs.
- All components targeting an UI control (or a UI element) rely on a URI.
- All components retrieving data from the UI can store the collected information in the CBTA execution context.
- Values stored in the execution context can be reused via the concept of tokens.

Components to Retrieve Data

Some components have the ability to collect information from the application UI, they are used to:

- Retrieve the value from properties or attributes of the targeted object.
- Check a value against the expected one
- Make the information available to subsequent components

Note
The term “Getter Component” is used when talking about components having the ability to retrieve data. They also have output parameters (at least one)

Input Parameters

URI

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

EXPECTED VALUE

In general, getter components have an ExpectedValue parameter. When this parameter is specified, the runtime library checks whether the retrieved value matches the expected one.

- If this is not the case, the test reports an error
- No check is performed when the ExpectedValue parameter is left empty
- The %blank% token must be used to check for an empty string

TARGET FIELD

Getter components may have a TargetField parameter. When this parameter is specified, the retrieved value is persisted to make it available to subsequent components. From the TargetField parameter value, a variable is created and its value is persisted in the execution context. The variable will be accessible by subsequent components via the concept of tokens. For more information on variables, see the section CBTA Execution Context.

Note that an %Output% token is always created (even when the TargetField parameter is left empty).
Output Parameter

**OUTPUT**

Getter components have an **Output** parameter which receives the collected information.

⚠️ Caution

*Do not confuse the `%Output%` tokens and the Output parameters.*

- Tokens have a limited scope. They can only be reused by the subsequent steps of the current test script.
- Output parameters are different. Their values are exposed at the test script level. They can be mapped to the input parameters of another test script using standard TCE features.

**Getter Components**

For SAP GUI transactions,

- `CBTA_GUI_GetProperty`

For SAP CRM applications,

- `CBTA_CRM_GetProperty`
- `CBTA_CRM_GetAttribute`

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

- `CBTA_WEB_GetProperty`
- `CBTA_WEB_GetAttribute`

**Getter Components to Check Application Messages**

- `CBTA_GUI_SB_GetMessageParams`
- `CBTA_CRM_A_GetMessageParams`
- `CBTA_WEB_A_GetMessage`
- `CBTA_UI5_A_GetMessage`
Components to Check Data

As mentioned, getter components can retrieve values and do some simple checks when their `ExpectedValue` parameter is set. For complex data consistency checks, the runtime library also concludes components that let you compare values using a Boolean operator.

**Note**

These components are the one used, by the generated test script, when checkpoints have been defined while recoding the scenario. For more details, see: CBTA – Test Recorder

---

**Input Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI](#) syntax on page 16.

**PROPERTY NAME (OR ATTRIBUTE NAME)**

This parameter specifies the name of the property (or the attribute) whose value is to be verified.

- The `exist` property is the one to use to check whether the target exists or not.
Note

CBTA discovers dynamically the control properties and attributes. The list of supported properties differs depending on the UI technology used by the application being tested.

Note that the component cannot check properties that return complex object types. For a complete list of property names, refer to the help file of the SAP GUI Scripting API.

**OPERATOR**

Specifies the boolean operator to use to compare the actual value with the expected one. Refer to the section “Checkpoint Operators” for more details.

**EXPECTED VALUE**

The expected value is the value that should be retrieved from the targeted control.

- The component will report an error if the value is not the expected one
- The check is not made if this parameter is empty
- Use the %blank% token to enforce the check against an empty value

**OPTIONS**

The options parameter enforces a type conversion before comparing the actual and expected values. Refer to the section “Checkpoint Options” for more details.

Additional options can influence the test behavior:

```
/x (exit) Interrupts the test when the comparison fails
```

**Output Parameter**

**OUTPUT**

These components have an Output parameter which receives the collected information. Note that the retrieved value is also stored in the CBTA Execution Context and can be reused via the %Output% token.

**Components to Check Data**

For SAP GUI transactions,

- CBTA_GUI_CheckProperty

For SAP CRM applications,

- CBTA_CRM_CheckProperty
- CBTA_CRM_CheckAttribute

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

- CBTA_WEB_CheckProperty
- CBTA_WEB_CheckAttribute
Checkpoint Operators

The operators supported by the CheckProperty and CheckAttribute components 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>
<tr>
<td>{contains}</td>
<td>Contains</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, the Test Recorder Wizard shows the list of the Boolean operators that you may use.

![Figure 6: Checkpoint Operators](image-url)
Regular Expressions

When using the \{matches\} operator the value must be expressed using the .NET regular expression syntax.

Recommendation

*Regular expressions can be checked online; a lot of websites provides this feature.*

Examples

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.number` Unit property exposed by a SAP UI5 tile, like in this example:

![Figure 7: Unit Property of a SAP UI5 tile](image-url)
Figure 8: Check using a Regular Expression
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 compared (new 3.0.8) Note that the VBScript DateValue function is used internally to perform the conversion.</td>
</tr>
<tr>
<td>/CC</td>
<td>Or /CustomConversion (new 3.0.8) The conversion is performed by calling the function GS_CustomConvert. 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)</td>
</tr>
</tbody>
</table>

Asynchronous Checkpoints

Additional options are available to define asynchronous checkpoints. For more details, refer to the CBTA – Test Recorder documentation.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/wsd(?)s</td>
<td>(start duration) 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 if the expected state is not met after the timeout.</td>
</tr>
</tbody>
</table>
The screenshot below shows the input parameters of a checkpoint waiting for a Web Dynpro Button to be enabled.

Figure 9: Input Parameters for an Asynchronous Checkpoint
3.3 Execution Context

The execution context can be seen as a shared memory where information is stored to make it available to the subsequent steps of the test script.

Variables

With CBTA, the execution context can be populated with computed values. The typical use case is to dynamically create a variable, in which the result of a step is stored and reuse by the next ones.

Example

For example, the business scenario could be to use the VA21 transaction to create a quotation, and use the ID of the newly-created quotation in the VA01 transaction to create a sales order.

This example can be automated using the CBTA_GUI_SB_GETMESSAGEPARAMS component. This component retrieves the parameter values of the status bar and stores them in variables that are prefixed by the transaction code.

For instance, the status bar of the VA21 transaction creates the following variables:

- VA21_MessageStatus
- VA21_MessageParameter0
- VA21_MessageParameter1

The next component retrieves the value from execution context by using the corresponding token as input parameter. The tokens matching the variables created by the VA21 transaction are:

- %VA21_MessageStatus%
- %VA21_MessageParameter0%
- %VA21_MessageParameter1%

The information retrieved from the status bar (of type GuiStatusbar) is visible in the execution report, as shown below:

![Tokens shown in the Execution Report](image)

The runtime library resolves the value of each component parameter by replacing each token with the value of the corresponding variable.
3.4 Tokens and Token Resolution

Tokens have been introduced to make it easy to share information between components. All component parameters (including the URI parameter) can use tokens. Each time a component starts, the runtime library looks for tokens, and replaces them with the value of the corresponding variable. The value is retrieved directly from the execution context.

Using tokens only makes sense when corresponding variable has been populated.

- This is typically done using a getter component such as the `CBTA_GUI.GetProperty` component or by defining checkpoints.
- You may also use the `CBTA_A_SetInExecutionCtxt` component to explicit create a variable and store its value

Some tokens are standard keywords that can dynamically retrieve information about the execution context. For instance:

- `%today%` - the current date
- `%yesterday%` - yesterday's date
- `%tomorrow%` - tomorrow's date
- `%random%` - a random number (6 digits)
- `%timestamp%` - returns a sortable timestamp value (format: YYYYMMDDHHMMSS)
- `%browser%` - the name of the current browser (new 3.0.9.3)

**OS Specific Tokens (new 3.0.8)**

Some tokens provide access to well-known OS Environment variables:

- `%userprofile%` - root folder to user-specific files
- `%appdata%` - location where user-specific files are stored
- `%computername%` - name of the local machine (where the test scripts are executed)
- `%username%` - name of the current user (used to logon to the Windows O.S.)
- `%temp%` - location where temporary files are stored

**SAPGUI-Specific Tokens**

SAP GUI scenarios support the following tokens:

- `%activeWindow%` - the index of the current window – 0 for the main window
- `%windowType%` - the type of the active window
- `%windowTitle%` - the window title
- `%screenNumber%` - the screen number of the current SAP GUI session
- `%transactionCode%` - the current transaction code
- `%sessionNumber%` - the current session number
VB Script Expressions

Token resolution also evaluates VB script expressions. This happens when the token starts with the equals sign “=”.

Example, the next week’s date:

- `%=Date+7%`

In this example the “Date+7” expression is evaluated using the VBScript Eval() function. This is flexible and makes it possible to evaluate any expressions.

Example, how to call the Weekday VB function:

- `%=Weekday(Date+2)%`

Example showing how several tokens can be used in a single input parameter:

- `%=Day("2010-02-16")%/=%=Month("2010-02-16")%/=%=Year("2010-02-16")%`

The previous example returns the date in the French date format:

- 16/02/2010

**Nested Tokens**

The token resolution can be nested using the $ character or the # character instead of the % character. Example showing different syntaxes that get the same information:

- `%=tomorrow%`
- `%=CDate(#today#)+1%`

Note that the two following syntaxes provide the same result but this is only a side-effect of implicit type conversions performed by the VBScript interpreter.

- `%=CDate(#today#)+1%`
- `%=CDate($today$)+1%`

The next section provides details about implicit conversion mechanism and their potential side-effects.

**Implicit Conversion, the Weak-Typing Pitfall**

As you may know, the VBScript language is a weakly-typed language. In other words, implicit conversions are performed when evaluating expressions and most of the time this is supposed to help the test engineer.

Unfortunately, implicit conversions are sometimes confusing and in some cases simple operations do not behave in a way that one would expect.

Here is an example:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Evaluation Result</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 + 2</td>
<td>3</td>
<td>Normal behavior</td>
</tr>
<tr>
<td>1 + &quot;2&quot;</td>
<td>3</td>
<td>Implicit conversion of the second operand to an integer</td>
</tr>
</tbody>
</table>
Implicit conversion of the first operand to an integer. This is an unexpected behavior (most of the other scripting languages do not perform such conversion).

Concatenation of two strings – the result being a string as well.

2 is lower than 10

A string starting with “2” is not lower than a string starting with “1”!

This implicit conversion mechanism has to be considered carefully when using tokens because the actual value of a token is always of type string. As a consequence, evaluating expressions for comparing two tokens may lead to unexpected results.

Example

Let’s assume that we have in our execution context two variables, a counter and an increment respectively named counter and step and that they both have their initial value set to 1. The table below shows the difference between using the #token# and the $token$ syntaxes.

<table>
<thead>
<tr>
<th>Expression Inside</th>
<th>Evaluation Result</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>%= #counter# + #step# %</td>
<td>2</td>
<td>Expected behavior.</td>
</tr>
<tr>
<td>%= $counter$ + $step$ %</td>
<td>11</td>
<td>Unexpected behavior - Concatenation of two strings. Result is also a string.</td>
</tr>
</tbody>
</table>

The token resolution handles differently the two syntaxes. The #token# syntax simply replaces the token by its current value. This leads to the evaluation below:

```
Eval(" 1 + 1 ")
```

The second one replaces the $token$ by a call to the InterpretToken function returning the actual value as a string. This leads to the evaluation to the following expression:

```
Eval( InterpretToken ("counter") + InterpretToken ("step") )
```

This being equivalent in the end to evaluating

```
"1" + "1"
```

Note that conversion can be performed explicitly by calling a function of the runtime library or any other VBScript functions.

Example using the ToInt() function:

```
%= ToInt($counter$) + ToInt($step$) %
```
Tokens for Escaping Characters

One may notice that the `percent` character has a specific meaning. As soon as two `percent` characters are detected, the fragment between them is considered as being the name of a variable or an expression. This may lead to some ambiguities and it might be necessary to escape some occurrences of the `percent` character to make sure they won’t be interpreted as a token.

Escaping the Percent Character

Your test script may, for instance, except the following value as input parameter:

```
10% - 20%
```

To make sure the two `percent` characters are not seen as a token, you must escape them using the following syntax:

```
10%percent% - 20%percent%
```

Checking for an Empty String

The `percent` character is not the only one that needs to be escaped. For instance, you may want to check that the actual value of an input field is empty. The token to use for such use case is the one below:

```
%blank%
```

Preserving Leading and Trailing Space Characters

They are situations where the leading and trailing space characters are important. The problem is that input parameters values are automatically trimmed when entered in the TCE UI. As a consequence, you may have to escape some space characters to preserve them.

Here is an example:

```
%space%-%space%
```

Passing with Multiline Values

Same issues with multiline values. There is no way to enter them in the TCE UI. The only option is to use a token like shown below.

```
First line.%crlf%Second line.
```
Here is the exhaustive list of character that you may need to escape:

- \%backslash\% – backslash (\) – It can avoid some conflicts when searching nodes in a tree.
- \%blank\% – an empty string – to distinguish a parameter not set (Null), from an explicitly empty string.
- \%cr\% – carriage return – CHR(13)
- \%crlf\% – consecutive CR and LF – CHR(13)+CHR(10)
- \%dollar\% – dollar (\$)
- \%lf\% – line feed – CHR(10)
- \%percent\% – \%
- \%quote\% – double quotes (") – note that simple quotes do not need to be escaped.
- \%space\% – space – to avoid trimming the parameter values.
- \%sharp\% – sharp (#)
- \%tab\% – CHR(9) – Tab character

**Escaping URI Input Parameters**

As you know, the [URI syntax](#) consists of several fragments and several URI attributes. The syntax is well defined and relies on specific separators. Here again we may have situations where the value of each URI attribute are to be escaped.

**Caution**

For some historical reasons the percent character cannot be used to escape tokens when used inside a URI. The dollar character must be used instead.

Example of URI using a counter to build the ID of an `<ANCHOR>` HTML element.

```plaintext
tag=A; id=toobar-button-$counter$
```

Example of URI searching for an `<ANCHOR>` with a multiline text.

```plaintext
tag=A; innerText=Fist line\n$crlf$\nSecond line
```

Example of URI searching for an `<ANCHOR>` where the text includes a semi-colon. Note that in this case we do not encode the semi-colon itself but the following space character.

```plaintext
tag=A; innerText=Fist part;$spaces$Second part
```
4 Keywords

4.1 Keywords for Checking Conditions

The following keywords are the ones to use to define conditional constructs.

- IF
- ELSE
- ENDIF

They can be used to only perform operations when a particular state or a specific execution context is met at runtime.

Example

The screenshot below shows a typical example where the three keywords are used to only perform a click when a particular UI element exists in the HTML content of the application being tested.

![Figure 11: CBTA Keywords - IF / ELSE / ENDIF]

Explanations

In this example the IF keyword checks whether the %EXISTENCE_OK% token is true or not before performing a mouse click using the CBTA_WEB_CLICK component. Of course, this only makes sense if the EXISTENCE_OK has been set beforehand using for instance the CBTA_WEB_GETPROPERTY component.

Keyword: IF

"IF" is a keyword. It provides the ability to check for a condition before executing some of the subsequent steps.

- This keyword must be used together with the ENDIF keyword.
The steps that are located between the IF and the ENDIF keywords are only performed when the condition is met.

The ELSE keyword can be used as well. The steps located after the ELSE keyword are only performed when the condition is NOT met.

The condition is defined by two operands and an operator which is used to compare them.

Information stored in the execution context can be checked using the token syntax.

Regular CBTA tokens like %today% can be used as well.

For instance, for a SAP GUI scenario, one may need to check whether a modal popup is being displayed before doing some actions. In such situation, the IF keyword can be used to check the actual value of the %activeWindow% token to somehow perform the equivalent of the following code:

```
IF %activeWindow% = 1 THEN
  // Do something
ELSE
  // Do something else
ENDIF
```

**Component Parameters**

**LEFTOPERAND**

Specifies the value of the left operand that is to be checked

**OPERATOR**

Specifies the boolean operator to use. Refer to the section “Checkpoint Operators” for more details.

**RIGHTOPERAND**

Specifies the value of the right operand that is to be compared with the left operand

**OPTIONS**

The options parameter lets you perform some adaptations or conversions before evaluating the condition. Refer to the section “Checkpoint Operators” for more details.

**Boolean Operators**

The operators supported are the ones below:

- = for "Equal to"
- < for "Less than"
- > for "Greater than"
- <= for "Less than or equal to"
- >= for "Greater than or equal to"
- <> for "Not equal to"
- {contains} for "Contains"
- {startsWith} for "Starts with"
- {endsWith} for "Ends with"
[matches] for checking whether the value matches a regular expression. The regular expressions are expressed using the .NET syntax.

**Options**

The options parameter lets you perform some adaptations or conversions of both the left and right operand before comparing them.

The supported options are:

- /u (for uppercase) - Both values are converted to upper-case before being compared
- /t (for trimmed) - Both values are trimmed before being compared
- /i (integer) - Both values are converted to an integer before being compared
- /f (float) - Both values are converted to a float (or double) before being compared
- /b (bool) - Both values are converted to a Boolean before being compared

**Keyword: ELSE**

“ELSE” is a keyword. It must be used between the IF and the ENDIF keywords. These keywords provide the ability to check for a condition before executing some of the subsequent steps.

- The steps that are located between the IF and the ELSE keywords are only performed when the condition is met.
- The steps located between the ELSE and the ENDIF keywords are only performed when the condition is **NOT** met.

**Keyword: ENDIF**

“ENDIF” is a keyword. It must be used after the IF keyword which provides the ability to check for a condition before executing some of the subsequent steps.

- The steps that are located between the IF and the ENDIF keywords are only performed when the condition is met.
- The ELSE keyword can be used as well. The steps located between the ELSE and the ENDIF keywords are only performed when the condition is **NOT** met.
4.2 DO / LOOP Iteration Keywords

The following keywords are the ones used to define loops and iterations.

- DO
- LOOP
- EXIT_DO

They can be used to iterate through the content of a table and thus perform operations for each and every row the table may contain at runtime.

Example

The screenshot below shows a typical example of a DO /LOOP iteration where the token %mycounter% is used to determine when to end the iteration.

![Figure 12: DO / LOOP Iteration Example](image-url)
Keyword: **DO**

**DO** is a keyword. It can be used to iterate over several steps. It defines where the loop starts.

- It must be used together with the **LOOP** keyword which defines where the loop ends.
- The **EXIT_DO** keyword must be used as well to determine when to stop the loop.

The **CounterName** parameter provides the name of the iteration counter. This counter is incremented automatically at runtime while iterating over the included steps. The actual value of the counter can be retrieve using the regular token syntax.

For instance, when **CounterName** is set to "index" its value can be reuse in the subsequent steps using %index% (or #index# for specific situations where the percent character is ambiguous).

Warning: Make sure to declare a different counter name when defining nested loops.

**Component Parameters**

**COUNTERNAME**
- Specifies the name of the iteration counter.

Keyword: **LOOP**

"**LOOP**" is a keyword. It must be used after the **DO** keyword. It ends the DO / LOOP structure and resume the execution flow to start the next iteration.

Keyword: **EXIT_DO**

**EXIT_DO** is a keyword. It must be used within a loop that has been defined using the **DO** and the **LOOP** keywords.

The **EXIT_DO** keyword interrupts the loop as soon as the condition is met.

A typical use case is to check the value of iteration counter that has been declared via the **CounterName** parameter of the **DO** keyword.

For instance, when **CounterName** is set to "index" its value can be checked using the %index% token.

**Component Parameters**

**LEFTOPERAND**
- Specifies the value of the left operand that is to be checked

**OPERATOR**
- Specifies the boolean operator to use. Refer to the section "Checkpoint Operators" for more details.

**RIGHTOPERAND**
- Specifies the value of the right operand that is to be compared with the left operand

**OPTIONS**
- The options parameter lets you perform some adaptations or conversions before evaluating the condition. Refer to the section "Checkpoint Operators" for more details.
4.3 FOR / NEXT Iteration Keywords

The following keywords are the ones used to iterate when the number of iterations is known in advance.

- FOR
- NEXT
- EXIT_FOR

They can be used to iterate through the content of a table and thus perform operations for each and every row the table may contain at runtime.

Examples

The screenshot below shows a typical example of a FOR / NEXT loop where the counter name is set to *mycounter*.

![Example Screenshot](image.png)

Figure 13: FOR / NEXT Iteration Example
Keyword: FOR

FOR is a keyword. It can be used to iterate over several steps. It defines where the loop starts.

- It must be used together with the NEXT keyword which defines where the loop ends.
- The EXIT_FOR keyword might be used as well to interrupt checking a specific condition.

The CounterName parameter provides the name of the iteration counter. This counter is incremented automatically at runtime while iterating over the included steps. The actual value of the counter can be retrieve using the regular token syntax.

For instance, when CounterName is set to "index" its value can be reuse in the subsequent steps using %index% (or #index# for specific situations where the percent character is ambiguous).

Warning: Make sure to declare a different counter name when defining nested loops.

Component Parameters

CounterName
- Specifies the name of the iteration counter

FROM
- Specifies the initial value of the iteration counter

TO
- Specifies the final value of the iteration counter

STEP
- Specifies the value being used when incrementing the iteration counter.

Keyword: NEXT

NEXT is a keyword. It defines the end of the loop and must be used together with the FOR keyword which defines where the loop starts.

The EXIT_FOR keyword might be used within the loop to interrupt the normal execution flow and stop iterating.

Keyword: EXIT_FOR

EXIT_FOR is a keyword. It must be used within a loop that has been defined using the FOR and the NEXT keywords.

The EXIT_FOR keyword interrupts the loop as soon as the condition is met.

A typical use case is to check the value of iteration counter that has been declared via the CounterName parameter of the FOR keyword.

For instance, when CounterName is set to "index" its value can be checked using the %index% token.
**Component Parameters**

**LEFTOPERAND**
- Specifies the value of the left operand that is to be checked

**OPERATOR**
- Specifies the boolean operator to use. Refer to the section "Checkpoint Operators" for more details.

**RIGHTOPERAND**
- Specifies the value of the right operand that is to be compared with the left operand

**OPTIONS**
- The options parameter lets you perform some adaptations or conversions before evaluating the condition. Refer to the section "Checkpoint Operators" for more details.
5 Components

5.1 Components Common to All UI Technologies

Some components can be used whatever the UI technology of the application being tested.

**CBTA_A_GetFromExecutionCtxt**

Technical Name: Actions\ExecutionContext\GetFromExecutionContext

This component copies information from the Execution Context and exposes it as an output parameter. This makes the information available to subsequent Test Scripts.

With this component the test engineer can, for instance, pass information from a CBTA test script to a QTP test script.

**Component Parameters**

**NAME**

This parameter specifies the name of the information in the Execution Context that the tester wants to retrieve and make available to subsequent Test Scripts.

**Output Parameter**

**OUTPUT**

This parameter is the output of the component. The subsequent steps of the test can use its value as input parameters.

**Explanations**

Checkpoints defined by the Check Picker, automatically populate the Execution Context with the value of the elements that are verified. A first execution of the test script might be necessary, to determine the name of the information stored in the Execution Context; the names are visible in the execution report.
CBTA_SetInExecutionContext

Technical Name: Actions\ExecutionContext\SetInExecutionContext

This component stores information in the Execution Context, and makes it available to subsequent components.

Component Parameters

NAME
This parameter specifies the name under which the information is stored. It can later on be used thanks to the concept of tokens. For instance, a name set to "myResult" will make the information available as %myResult% in the subsequent step of the Test Script.

VALUE
This parameter specified the value that is to be stored in the Execution Context.
The "ReportMessage" component troubleshoots complex scenarios by adding custom messages to the execution report.

**Component Parameters**

**SEVERITY**
The severity of the message – INFO, WARNING, DONE, FAILED

**TOPIC**
The Topic parameter is a short text to categorize the message.

**MESSAGE**
Any message can be specified here.

**OPTIONS**
Reserved for future use

**Notes**

Like all component parameters, the **MESSAGE** parameter can use tokens to retrieve the value of a variable from the CBTA Execution Context.

For instance, one could include the ID of an order created by transaction VA01, in a message, using the message below:

“Standard order %VA01_MessageParameter1% has been created”
The "CompareValues" component compares two values by evaluating a Boolean expression. The test fails and reports an error when the expression evaluation returns false. The component also captures a screenshot of the current screen when the test fails.

**Component Parameters**

**SAPVALUE1**

“SAPVALUE1” specifies the first value; the left-hand operand of the expression to evaluate.

The value specified here will typically be a token that has been previously set using a “getter” component; a component expecting a TargetField parameter such as the GetProperty and the GetMessageParameter component.

**SAPVALUE2**

“SAPVALUE2” specifies the second value; the right-hand operand of the expression to evaluate.

**COMPARISONOPERATOR**

The operator must specify a Boolean operator to compare the two operands. Refer to the section “Checkpoint Operators” for more details.

**OPTIONS**

The options parameter enforces a type conversion before the comparison. Refer to the section “Checkpoint Options” for more details.
CBTA_A_SetCondition (deprecated)

Technical Name: Actions\Verifications\SetCondition

This component is deprecated. Use the IF / ELSE / ENDIF keywords instead.

The "SetCondition" component declares a condition. The condition will be used by subsequent test components to determine whether they have to perform their operation.

This component is deprecated. SAP recommends using keywords instead. The following keywords are available:

- IF
- ELSE
- ENDIF

For more information, refer to the section Keywords as Default Components.

Component Parameters

NAME
A name to identify the condition – any string can be used here

SAPCONDITION
SAPCONDITION is an expression to be checked by subsequent test components.

OPTIONS
- /o (only once) – the condition will be checked only once.

Expression Evaluation

The SAPCONDITION parameter provides a boolean expression which can be evaluated by the VB script interpreter.

For example, one could declare a condition to check whether the current window is a popup, and thus only perform the subsequent actions when the condition is met. A typical use case is to exit from a popup window which appears from time to time, depending on some external application events.

Postponing Token Evaluation

Like all component parameters, the SAPCONDITION parameter can use tokens to retrieve the value of a variable from the CBTA Execution Context. However, the token is resolved only once, when setting the condition, and not when checking it. This is not what is expected when setting conditions. To postpone the token interpretation, the SAPCONDITION parameter supports an alternative syntax, where the dollar character is used (instead of the percent character) to escape the tokens.

Example:

- %activeWindow%=1 – evaluation is performed only once, when setting the condition
- $activeWindow$=1 – evaluation is postponed until the condition check. The token is evaluated (interpreted) each time the condition is used.
Expression Persistence

All declared conditions have a name. They are persisted in the CBTA execution context, and by default made available during the entire test. The CBTA_A_RemoveCondition component removes a condition.

The OPTIONS parameter can be set to “/o” to automatically remove the condition from the next component execution.

CBTA_A_RemoveCondition (deprecated)

Technical Name: Actions\Verifications\RemoveCondition

This component is deprecated. Use the IF / ELSE / ENDIF keywords instead.

This component removes a previously declared condition, using the CBTA_A_SetCondition component.

Component Parameters

NAME

The name of a previously declared condition

OPTIONS

/a (all) – when the name is omitted, this option removes all conditions.

CBTA_A_Wait

Technical Name: Actions\Wait

The “Wait” component can make the test pause for a certain time before proceeding.

Component Parameters

THEVALUE

Specifies the waiting time in milliseconds.

OPTIONS

Reserved for future use

Note

CBTA relies on a built-in wait mechanism at runtime. In other words, the test player automatically waits for the application to be ready before performing the next actions. As a consequence, the CBTA_A_WAIT component is normally useless.

There are scenarios where the implicit wait mechanism is not sufficient. In such situation, the recommendation is the following:

- Use Asynchronous Checkpoints to wait for a job running in background
- Add an additional wait time directly in the URI using the Wait and Attempts URI attributes.
6 SAP GUI Transactions

CBTA delivers components to automate the testing of SAP transactions that are built using the SAPGUI UI Technology. The components supporting this technology are prefixed using “CBTA_GUI_”.

6.1 Identifying SAP GUI Controls

The SAP GUI components targeting a UI controls have a URI parameter which provides the information required to find the target. The URI syntax is flexible enough to allow searching for the control using various criteria.

**URI Syntax for SAP GUI**

The URI is composed of key-value pairs, separated by a semicolon and a space character.

For example:

```
label=<controlLabel>; type=<controlType>; id=<controlId>
```

A typical URI provides the ID of the control and its type.

- The ID is used to search for the control using the official SAP GUI Scripting API.
- The type ensures that the object found is the expected one. If this is not the case, the test reports an error.
- Additional information is not mandatory. To improve the feedback in the execution report when an error occurs during the test, the label can be specified.

As an alternative to the ID, the URI can provide the name of the control, as shown below:

```
label=<controlLabel>; type=<controlType>; name=<controlName>
```

This is not the recommended way to identify a control, because names may conflict. The syntax is only supported for backward compatibility.

**Examples of valid URIs**

```
label=Main Window; type=GuiMainWindow; id=/app/con[3]/ses[0]/wnd[0]
label=Main Window; type=GuiMainWindow; id=wnd[0]
```

As shown in these examples, the ID can be relative, but a full ID (with information about the connection and the session) is also supported by the runtime library.

**URI Syntax with Text**

For complex scenarios it might be necessary to search for a control using the information it displays, its Text property. In that case, the URI must provide the text of the control, as shown below:

```
type=<controlType>; text=<controlText>
```

This is not the recommended way to identify a control, because the test will depend on the user language.
URI Syntax with Text and Index

When several controls have the same text, it might be necessary to specify the index of the control, to avoid ambiguities.

\[
\text{type}=<\text{controlType}>; \text{text}=<\text{controlText}>; \text{index}=<\text{controlIndex}>
\]

If the index is not specified, its default value is 0.

URI Syntax with Label and Text:

The label and the text are not the same thing. In the example below, the label is only used to improve the feedback in the execution report. When searching for an input field (of type \texttt{GuiTextField}), the text must provide the value of the field, not its label.

\[
\text{Label}=<\text{aLabel}>; \text{type}=<\text{controlType}>; \text{text}=<\text{controlText}>
\]
6.2 SAP GUI - Action Components

**CBTA_GUI_A_CaptureScreen**

Technical Name: Actions\CaptureScreen

The "CaptureScreen" component captures a screenshot of the active window of the SAP GUI session.

- The persisted screenshot will be visible in the execution report.

**Component Parameters**

**OPTIONS**

Reserved for future use

**Known Limitations**

The screenshot is captured by the Hardcopy method of the SAP GUI Scripting API. This method does not work when the computer is locked or the screen-saver is active.

**CBTA_GUI_A_CheckTransactionCode**

Technical Name: Actions\CheckTransactionCode

The "CheckTransactionCode" component checks whether the current transaction is the one expected by the scenario being tested.

The check result is in the execution report.

- The test status is "PASSED" when the transaction is the correct one
- Otherwise the test status is "FAILED"

**Component Parameters**

**EXPECTED_TRANSACTION_CODE**

Specifies the expected transaction code.

**OPTIONS**

- /x (exit) – Interrupts the test execution when the current transaction is not the expected one
- /c (capture) – Captures a screenshot of the active window if the transaction is not the expected one

**CBTA_GUI_A_CloseWindow**

Technical Name: Actions\CloseWindow

The "CloseWindow" component closes the specified window. It emulates a mouse click on the button to close a popup window.

**Component Parameters**

**URI**

Specifies the URI of the targeted window. For more information, refer the URI syntax on page 16.
OPTIONS
Reserved for future use.

CBTA_GUI_A_EndTransaction

Technical Name: Actions\EndTransaction

The “EndTransaction” component ends the current transaction.

Component Parameters

OPTIONS
Reserved for future use.
**CBTA_GUI_ExecuteStatement**

Technical Name: Actions\ExecuteStatement

The “ExecuteStatement” component calls custom functions (or subroutines). It can address situations, such as dynamic scenarios, that the default components do not support.

The advantage is that the tester can put the custom code in a dedicated library and call it directly. There is no need to create a new component.

### Component Parameters

**LIBRARY**
The Library parameter is the relative path, from the CBASE folder, of the library which contains the statement to execute.

**STATEMENT**
The Statement parameter provides the instruction to be executed.

**OPTIONS**
Reserved for future use.

**Note**

- The statement specified will typically invoke a subroutine or a function and will be executed by the VB script interpreter. Ensure the syntax of the statement is correct.
- Like all default components, the **STATEMENT** parameter can use tokens to retrieve the value of a variable from the **Execution Context**.

### Examples of Valid Statements

The examples below explain how to invoke a subroutine.

Example logging “Hello World” (using the CBTA Helper Class):

```plaintext
CBTA.Log "Hello World"
```

Example using a token to log the current transaction code:

- Quotes are required because the Log subroutine expects a string as input parameter.

  ```plaintext
  CBTA.Log "%transactionCode%"
  ```

Example using a variable to log the current transaction code:

- The two lines below are equivalent. The quotes have no effect in either case.

```plaintext
CBTA.Log "%transactionCode%"
```
CBTA\Gui\Invoke\Function

Technical Name: Actions\InvokeFunction

The “InvokeFunction” component is similar to the “ExecuteStatement” component, and it calls custom functions. It addresses some situations, such as dynamic scenarios, that the other default components do not support.

With this component, the tester can put the custom code in a dedicated library and call it directly.

Component Parameters

/library

The Library parameter is the relative path, from the CBASE folder, of the library which contains the statement to execute.

/FunctionName

The FunctionName parameter provides the name of the function to be executed. The function called using this component must have five parameters.

- PARAMETER1
- PARAMETER2
- PARAMETER3
- PARAMETER4
- OPTIONS

Output Parameter

The component has an Output parameter that receives the value returned by the custom function. This value can be used by subsequent components in the test.

Explanations:

The component resolves each token before passing the parameter value to the custom function, so you can pass the date using the %today% token in any of the parameters.

The custom function receives the parameter value Null if a parameter is empty. The %blank% token passes an empty string as input parameter.
**CBTA_GUI_A_LaunchAndLogin**

Technical Name: Actions\LaunchAndLogin

The **LaunchAndLogin** component initializes the SAP GUI session to be used by all subsequent components.

**Component Parameters**

**SAPSYSTEM**

"SAPSYSTEM" specifies the SAP system to connect to. The component checks the SAPLOGON configuration for a matching entry, and uses this data to create the connection.

**SAPCLIENT**

Specifies the client number

**SAPUSER**

Specifies the name of the login user

**SAPPASSWORD**

"SAPPASSWORD" specifies the password of the user. Make it secure using the QTP Password Encoder.

**SAPLANGUAGE**

"SAPLANGUAGE" specifies the preferred language; for example, EN (for English) or JA (for Japanese). Test execution using QTP is sensitive to language-specific data formats. Use the language of the data in the DataTable.

**SAPOPTIONS**

/r (resize) – Resizes the main window according to CBASE configuration.

**CBTA_GUI_A_LogOff**

Technical Name: Actions\LogOff

The **LogOff** component logs off from the SAP System. This ends the SAP GUI session.

**Component Parameters**

**OPTIONS**

Reserved for future use

**CBTA_GUI_A_PressKey**

Technical Name: Actions\PressKey

The **PressKey** component emulates a key press by a user while.
Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted window. For more information, see the URI syntax on page 16.

**KEY**
KEY specifies the key code pressed. The key code can be a numeric value or a string. The recommendation to find the appropriate value is to record the scenario.

**OPTIONS**
Reserved for future use

**Key Codes**
The list below shows the mapping between the numeric key code and its equivalent string. Both formats can be used without any side-effects. The string representation of the key does not depend on the user language.

- 0 Enter
- 1 F1
- 2 F2
- 3 ...
- 9 F9
- 10 F10
- 11 F11
- 12 ESC
- 13 Shift+F1
- 14 Shift+F2
- 15 ...
- 24 Shift+F12
- 37 Ctrl+Shift+F1
- 38 Ctrl+Shift+F2
- 39 ...

**CBTA_GUI_A_StartTransaction**

Technical Name: Actions\StartTransaction

The StartTransaction component starts a transaction. The session should have been started using the LaunchAndLogin component. If this is not the case, the operation is performed against the first available SAPGUI session.

Component Parameters

**SAPTRANSACTIONCODE**
SAPTRANSACTIONCODE is the code of the transaction to be started.

**OPTIONS**
Reserved for future use
6.3 SAP GUI - Generic Components for SAP GUI

CBTA_GUI_CheckProperty

Technical Name: Controls\CheckProperty

The CheckProperty component is a generic component to retrieve and check the value of the properties exposed by SAP GUI Scripting controls. It is inserted in the test automatically when a checkpoint is defined while recording a scenario.

Component Parameters

URI

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

PROPERTYNAME

This parameter specifies the name of the property whose value is to be retrieved. The property names that are commonly used are:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Returns the text or the value displayed by the control, depending on its nature</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the control</td>
</tr>
<tr>
<td>exist</td>
<td>Special property used to check whether the control is displayed or not.</td>
</tr>
</tbody>
</table>

Note that the component cannot check properties that return complex object types. For a complete list of property names, refer to the help file of the SAP GUI Scripting API.

OPERATOR

The operator is a boolean operator to compare the actual value with the expected one. Refer to the section “Checkpoint Operators” for more details.

EXPECTEDVALUE

The expected value is the value that should be retrieved from the targeted control.

- The component will report an error if the value is not the expected one
- The check is not made if this parameter is empty
- Use the %blank% token to enforce the check against an empty value

OPTIONS

The options parameter enforces a type conversion before comparing the actual and expected values. Refer to the section “Checkpoint Options” for more details.

Additional options can influence the test behavior:

/x (exit) | Interrupts the test when the comparison fails
CBTA_GUIGetProperty

Technical Name: Controls\GetProperty

The GetProperty component is a generic component to retrieve the value of the properties of the targeted SAP GUI objects.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

PROPERTYNAME
This parameter specifies the name of the property whose value is to be retrieved.

The component cannot check properties that return complex object types. For a complete list of property names, refer to the help file of the SAP GUI Scripting API.

EXPECTEDVALUE
The expected value

TARGETFIELD
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

Output Parameter

OUTPUT
This component has an output parameter providing the value of the property. The subsequent components can use its value as input parameters.

Note
This component has standard behavior. See section Getter Components for more details.
**CBTA_GUI_GetText**

Technical Name: Controls\GetText

The “GetText” generic component gets the value of the targeted SAP GUI objects. It is similar to the Controls\GetProperty component when “Text” is specified as PropertyName.

**Component Parameters**

**URI**
Specifies uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**EXPECTEDVALUE**
The expected value of the Text property.

**TARGETFIELD**
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Output Parameter**

**OUTPUT**
This component has an output parameter to receive the retrieved value. Subsequent components can use its value as input parameters.

**Note**
This component has standard behavior. See section Getter Components for more details.
**CBTA_GUI_SelectContextMenuItem**

Technical Name: Controls\SelectContextMenuItem

The “SelectContextMenuItem” component emulates selecting an item in the context menu of the target tree. The component for opening the menu does not exist; this operation is implicit at runtime when executing a test.

Component Parameters

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**FUNCTIONCODE**

The menu item to be selected.
CBTA_GUI_SetFocus

Technical Name: Controls\SetFocus

The “SetFocus” generic component puts the focus on the targeted SAP GUI objects.

- This operation is normally not necessary because most of the SAP GUI Scripting operations do not require that the targeted control gets the focus first.
- However, some coding (on the server side) may check whether the focus was set properly. In that situation, an explicit invocation of the SetFocus method might be necessary to make the test run properly.

Component Parameters

URI

The uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.
CBTA_GUI_SetProperty

Technical Name: Controls\SetProperty

“SetProperty” is a generic component. It changes the value of properties exposed by the SAP GUI Scripting Objects. It does not work when targeting read-only properties.

Component Parameters

URI
The uniform resource identifier of the targeted object. For more information, see the URI syntax on page 16.

PROPERTYNAME
The name of the public property exposed by the targeted object.

The public properties available depends on the type of control targeted. Refer to the official SAP GUI Scripting help (delivered with the SAP Front End) for the complete list of public properties.

VALUE
The new value of the property.

Known Limitations
The value must have the format the targeted property expects.

The runtime library does not check the format. For example, if the targeted property expects a date, the input parameter must provide the date in the appropriate format.
CBTA_GUISetText

Technical Name: Controls\SetText

The “SetText” generic component sets the value of the Text property of the targeted control. This is similar to the SetProperty component when the PropertyName is “Text”.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

THEVALUE
Specifies the new value of the targeted control; the new value of the Text property.

Known Limitations

The value must match the format of the targeted SAP GUI control. The runtime library does not check the format. For example, if the targeted control expects a date, the input parameter must provide the date in the appropriate format.

Miscellaneous

This component is used by the Process Flow Analyzer. Text field are commonly assigned using this component. Assigning the cell value in a table control also relies on this component.

See Also

If the targeted control expects a Boolean value, it is preferable to use the dedicated component which converts the input value appropriately.

Examples:

- GuiCheckbox/Selected to change the state of a checkbox.
- GuiRadioButton/Selected to change the state of a radio button.
6.4 SAP GUI – Control Components for SAP GUI

**CBTA_GUI BTN_PressButton**

Technical Name: Controls\GuiButton\Press

The “Press” component emulates a click on the targeted button.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the target GuiButton object. For more information, refer to the **URI** syntax on page 16.

**CBTA_GUI_CB_GetSelected**

Technical Name: Controls\GuiCheckBox\GetSelected

The “GetSelected” component gets the state of the targeted checkbox.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**EXPECTEDVALUE**

Specifies the expected state (True or False).

**TARGETFIELD**

This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to the **Getter Components** section.

**Component Output**

**OUTPUT**

This component has an output parameter which receives the checkbox state. The subsequent components can use its value as input parameters.

**Note**

This component has standard behavior. See section **Getter Components** for more details.
CBTA_GUI_CB_SetSelected

Technical Name: Controls\GuiCheckBox\SetSelected

The "SetSelected" component changes the state of the targeted checkbox.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

THEVALUE
Specifies the new state of the checkbox (true or false).

Note
No operation is performed if the value is empty.

CBTA_GUI_CB_GetKey

Technical Name: Controls\GuiComboBox\GetKey

The “GetKey” component gets the key of the selected entry.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, see the URI syntax on page 16.

EXPECTEDVALUE
Specifies the expected value of the key.

TARGETFIELD
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

Component Output

OUTPUT
This component has an output parameter which receives the key. The subsequent components can use its value as input parameters.

Note
This component has standard behavior. See section Getter Components for more details.
CBTA_GUI_CB_GetValue

Technical Name: Controls\GuiComboBox\GetValue

The “GetValue” component gets the value of the selected entry.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

EXPECTED_VALUE
Specifies the expected value.

TARGET_FIELD
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

Component Output

OUTPUT
This component has an output parameter which receives the combobox value. The subsequent components can use its value as input parameters.

Note
This component has standard behavior. See section Getter Components for more details.

CBTA_GUI_CB_SetKey

Technical Name: Controls\GuiComboBox\SetKey

The “SetKey” component selects one of the combobox entries

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

THE_VALUE
Specifies the key of the entry to be selected.

CBTA_GUI_CB_SetValue

Technical Name: Controls\GuiComboBox\SetValue

The “SetValue” component selects one of the combobox entries.
This component is not recommended because the behavior is language-dependent. The **GuiCombobox\SetKey** component should be used instead.

### Component Parameters

**Uri**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI](#) syntax on page 16.

**TheValue**

Specifies the value of the entry to be selected.
CBTA_GUI_GV_ClearSelection

Technical Name: Controls\GuiGridView\ClearSelection

The "ClearSelection" component clears the selection of the targeted grid.

Component Parameters

Uri

Specifies the uniform resource identifier of the targeted control. For more information, refer to the URI syntax on page 16.

CBTA_GUI_GV_ClickCurrentCell

Technical Name: Controls\GuiGridView\ClickCurrentCell

The "ClickCurrentCell" component emulates a mouse click on the current cell of the target grid.

Component Parameters

Uri

Specifies the uniform resource identifier of the targeted control. For more information, refer to the URI syntax on page 16.

Note

The behavior of this component depends on the state of the target control. Set the current cell first, using the GuiGridView\SetCurrentCell component.

CBTA_GUI_GV_DeleteRows

Technical Name: Controls\GuiGridView\DeleteRows

The “DeleteRows” component deletes rows from the targeted grid.

Component Parameters

Uri

Specifies the uniform resource identifier of the target control. For more information, refer to the URI syntax on page 16.

Rows

Specifies the (zero-based) row number that is to be removed from the grid.

This parameter also supports a comma-separated string of indexes or index ranges; for example, “3, 5-8, 14, 15”. The indexes must be ordered and should not overlap.

For more information on how to use index ranges, refer to the official SAP GUI Scripting help delivered with the SAP Front End.
**CBTA_GUI_GV_DeselectColumn**

Technical Name: Controls\GuiGridView\DeselectColumn

This component removes the specified column from the set of selected columns.

*Component Parameters*

**URI**

Specifies the uniform resource identifier of the target control. For more information, refer to the [URI syntax](#) on page 16.

**COLUMN**

Specifies the name of the column.

**CBTA_GUI_GV_DoubleClickCell**

Technical Name: Controls\GuiGridView\DoubleClickCurrentCell

The “DoubleClickCurrentCell” component emulates a mouse double-click on the current cell of the target grid.

*Component Parameters*

**URI**

Specifies the uniform resource identifier of the target control. For more information, refer to the [URI syntax](#) on page 16.

*Note*

The behavior of this component depends on the state of the target control. Set the current cell first, using the [GuiGridView\SetCurrentCell](#) component.

**CBTA_GUI_GV_DuplicateRows**

Technical Name: Controls\GuiGridView\DuplicateRows

The “DuplicateRows” component duplicates some rows of the target grid.

*Component Parameters*

**URI**

Specifies the uniform resource identifier of the target control. For more information, refer to the [URI syntax](#) on page 16.

**ROWS**

Specifies the (zero-based) row number that is to be duplicated.
Advanced Use Cases

The **Rows** parameter also supports a comma-separated string of indexes or index ranges; for example, "3, 5-8, 14, 15". If a range of indexes is duplicated, all the new lines are inserted as one block, before the old lines. The indexes must be ordered and should not overlap.

For more information on how to use index ranges, refer to the official SAP GUI Scripting help delivered with the SAP Front End.

Example of a grid containing two rows:

<table>
<thead>
<tr>
<th>0</th>
<th>Value A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Value B</td>
</tr>
</tbody>
</table>

If rows is "0,1" then the resulting table would be:

<table>
<thead>
<tr>
<th>0</th>
<th>Value A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Value A</td>
</tr>
<tr>
<td>2</td>
<td>Value B</td>
</tr>
<tr>
<td>3</td>
<td>Value B</td>
</tr>
</tbody>
</table>

If on the other hand rows is "0-1" then the resulting table is:

<table>
<thead>
<tr>
<th>0</th>
<th>Value A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Value B</td>
</tr>
<tr>
<td>2</td>
<td>Value A</td>
</tr>
<tr>
<td>3</td>
<td>Value B</td>
</tr>
</tbody>
</table>
CBTA_GUI_GV_FindRow

Technical Name: Controls\GuiGridView\FindRowByContent

This component searches for one or several rows checking the cell content of a particular column. Starting with SP05, scrolling the grid content is implicit.

Component Parameters

URI
Specifies the uniform resource identifier of the parent grid (of type GuiGridView). For more information, refer to the URI syntax on page 16.

COLUMNTITLE
Specifies the title of the column. The title is the information visible to the end user.

OPERATOR
The operator is a Boolean operator to compare the actual value with the expected one. Refer to the section “Checkpoint Operators” for more details.

CELLCONTENT
Specifies the value to search for.

OPTIONS
There are various ways to search for the row. The options below define the action to perform once the rows have been found.

Note: Options defining actions start with a capital letter.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Select</td>
<td>When this option is specified the first row matching the criteria is selected. This to avoid having to select the row using the CBTA_GUI_GV_SELECTROW component.</td>
</tr>
<tr>
<td>/Select</td>
<td>When these options are specified all rows matching the criteria are selected.</td>
</tr>
<tr>
<td>/Multiple</td>
<td></td>
</tr>
<tr>
<td>/Quiet</td>
<td>(new 3.0.8) An execution error is reported by default when no row matches the criteria. The /Quiet option can be used to get rid of the error and get an INFO message added to the execution report instead.</td>
</tr>
</tbody>
</table>
Type Conversion Options

Some other options can be used to alter or convert both the actual cell value and the expected cell value before comparing them.

<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 (or 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>

Component Output

**OUTPUT**

This component has an output parameter receiving the row number or a comma separated list of row numbers.

The subsequent steps may rely on either the output parameter or the %Output% token to reuse the information.
**CBTA_GUI_GV_GetCellChecked**

Technical Name: Controls\GuiGridView\GetCellCheckBoxChecked

The “GetCellCheckBoxChecked” component gets the state of a checkbox in a grid. The regular “GuiCheckBox\GetSelected” cannot be used here because additional information is required to find a check box control embedded within a GuiGridView control.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted control. For more information, refer to the URI syntax on page 16.

**ROW**

Specifies the row number of the checkbox. The row number property starts at zero.

**COLUMN**

Specifies the column name of the checkbox.

**EXPECTED_VALUE**

Specifies the expected state (True or False).

**TARGET_FIELD**

This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**

This component has an output parameter which receives the checkbox state. The subsequent components can use its value as input parameters.

**Note**

This component has standard behavior. See section Getter Components for more details.
**CBTA_GUI_GV_GetCellState**

Technical Name: Controls\GUI\GridView\GetCellState

The “GetCellState” component retrieves the state of a cell in a grid. According to the SAP GUI Scripting documentation, the possible values are:

- Normal
- Error
- Warning
- Info

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted control. For more information, refer to the URI syntax on page 16.

**ROW**
Specifies the row number of the cell

**COLUMN**
Specifies the column name of the cell

**EXPECTED VALUE**
Specifies the expected state

**TARGET FIELD**
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**
This component has an output parameter which receives the cell state. The subsequent components can use its value as input parameters.

**Note**
This component has standard behavior. See section Getter Components for more details.
**CBTA_GUI_GV_GetCellValue**

Technical Name: Controls\GuiGridView\GetCellValue

The “GetCellValue” component gets the value of a cell in a grid.

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted control. For more information, refer to the URI syntax on page 16.

**ROW**
specifies the row number of the cell.

**COLUMN**
Specifies the column name of the cell.

**EXPECTED_VALUE**
Specifies the expected state.

**TARGETFIELD**
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**
This component has an output parameter which receive the cell value. The subsequent components can use its value as input parameters.

**Note**
This component has standard behavior. See section Getter Components for more details.
CBTA_GUI_GV_InsertRows

Technical Name: Controls\GuiGridView\InsertRows

The “InsertRows” component inserts rows in the target grid.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted control. For more information, refer to the URI syntax on page 16.

ROWS
Specify a comma-separated text of indexes or index ranges; for example, “3, 5-8, 14, 15”. A new row will be added at the given index, moving the old row one line down. If a range of indexes is inserted, all the new lines are inserted as one block, before any of the old lines. The entries must be ordered and not overlap, otherwise an exception is raised.

For more information on how to use index ranges, refer to the official SAP GUI Scripting help delivered with the SAP Front End.

CBTA_GUI_GV_ModifyCell

Technical Name: Controls\GuiGridView\ModifyCell

The “ModifyCell” component modifies the value of a cell in a grid.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

ROW
Specifies the row number of the cell.

COLUMN
Specifies the column name of the cell.

THEVALUE
Specifies the new value of the targeted cell.
CBTA_GUI_GV_ModifyCheckBox

Technical Name: Controls\GuiGridView\ModifyCheckBox

The “ModifyCheckBox” component modifies the state of a checkbox in a grid.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI syntax](#) on page 16.

**ROW**
Specifies the row number of the cell.

**COLUMN**
Specifies the column name of the cell.

**CHECKED**
Specifies the new checkbox state (True or False).

CBTA_GUI_GV_MoveRows

Technical Name: Controls\GuiGridView\MoveRows

The “MoveRows” component moves rows within a grid.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI syntax](#) on page 16.

**FROMROW**
Specifies the index of the first row to move.

**TOROW**
Specifies the index of the last row to move.

**DESTROW**
Specifies where to move the selected rows to.
**CBTA_GUI_GV_PressButton**

Technical Name: Controls\GuiGridView\PressButton

This component emulates a mouse click on a button in a cell. The component reports an error if the targeted cell does not contain a button.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**ROW**

Specifies the row number of the cell.

**COLUMN**

Specifies the column name of the cell.

---

**CBTA_GUI_GV_PressColumnHeader**

Technical Name: Controls\GuiGridView\PressColumnHeader

This component emulates a mouse click on the header of the column.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**COLUMN**

Specifies the column name. The component reports an error if the column parameter does not identify a valid column.

---

**CBTA_GUI_GV_Press_Enter**

Technical Name: Controls\GuiGridView\PressEnter

This component emulates pressing the Enter key.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, see the URI syntax on page 16.
**CBTA_GUI_GV_Press_F1**

Technical Name: Controls\GuiGridView\PressF1

This component emulates pressing the F1 key.

*Component Parameters*

*Uri*

Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI](#) syntax on page 16.

**CBTA_GUI_GV_Press_F4**

Technical Name: Controls\GuiGridView\PressF4

This component emulates pressing the F4 key.

*Component Parameters*

*Uri*

Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI](#) syntax on page 16.

**CBTA_GUI_GV_TB_PressButton**

Technical Name: Controls\GuiGridView\PressToolbarButton

This component emulates a mouse click on a button in the toolbar of a grid, not the grid itself, like the “PressButton” component.

*Component Parameters*

*Uri*

Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI](#) syntax on page 16.

*Id*

The button ID
**CBTA_GUI_GV_TB_PressMenuItem**

Technical Name: Controls\GuiGridView\PressToolbarContextMenuItem

The "PressToolbarContextMenuItem" component emulates selecting an item in the context menu of a button in the toolbar attached to the targeted grid.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**BUTTONID**

Specifies the button ID.

**FUNCTIONCODE**

Specifies the menu item to be selected.

By default, the component expects the code of the context menu item. The text can be specified instead, by setting the Options parameter to /pt.

**OPTIONS**

- /pt – Use this option when the **FUNCTIONCODE** parameter, not the code of the context menu item, provides the text.

**Recommendations**

- Use the Process Flow Analyzer to determine the expected values of the **BUTTONID** and the **FUNCTIONCODE**.
- CBTA Object Spy also provides this information, but it might be difficult to find it in a complex UI.

**CBTA_GUI_GV_SelectAll**

Technical Name: Controls\GuiGridView\SelectAll

The "SelectAll" component selects the whole grid content (all rows and all columns).

**Component Parameters**

**URI**

The uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**CBTA_GUI_GV_SelectColumn**

Technical Name: Controls\GuiGridView\SelectColumn

The "SelectColumn" component selects the specified column.
Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**COLUMN**
Specifies the column name. The component reports an error if the column parameter does not identify a valid column.

---

**CBTA_GUI_GV_SelectMenuItem**

Technical Name: Controls\GuiGridView\SelectContextMenuItem

The "SelectContextMenuItem" component emulates selecting an item in the context menu of the target grid. The component to open the menu does not exist; this operation is implicit at runtime when executing a test.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**FUNCTIONCODE**
Specifies the code of the context menu item to be selected.

By default, the component expects the code of the context menu item. The text can be specified instead, by setting the Options parameter to /pt.

**OPTIONS**
- /pt – Use this option when the FunctionCode parameter provides the text, instead of the code of the context menu item.

**Note**
- Use the Process Flow Analyzer to determine the expected values of the FunctionCode.
- The SAP Object Spy also provides the information, but might be difficult to find it in a complex UI.

---

**CBTA_GUI_GV_SetCurrentCell**

Technical Name: Controls\GuiGridView\SetCurrentCell

The "SetCurrentCell" component sets the current cell in a grid.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.
**Row**
Specifies the row number of the cell.

**Column**
Specifies the column name of the cell.

**Note**
If row and column identify a valid cell, this cell becomes the current cell. Otherwise, an exception is raised.

---

**CBTA_GUI_GV_SetSelectedRows**

Technical Name: Controls\GuiGridView\SetSelectedRows

The “SetSelectedRows” component selects rows in a grid.

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**THE_VALUE**
Specifies the index of the row to select.

**Advanced Use-cases**

The Rows parameter also supports a comma-separated string of indexes or index ranges; for example, “3, 5-8, 14, 15”.

For more information on how to use index ranges, refer to the official SAP GUI Scripting help delivered with the SAP front end.
CBTA_GUI_HV_StartWebController

Technical Name: Controls\GuiHTMLViewer\StartWebController

SAP GUI transactions have the capacity to embed some HTML content using the GuiHTMLViewer control. The test automation of the embedded content is challenging because:

- Actions made against the embedded HTML content embedded in SAP GUI Transactions are not recorded by default.
- A manual adaptation of the generated test script might be necessary.

Even though this kind of scenarios cannot be recorded, the execution is now possible thanks to this component. It basically provides the ability to attach the test player to the embedded Internet Explorer session.

**Component Parameters**

**Uri**

Specifies the uniform resource identifier of the targeted GuiHTMLViewer control. This information can be retrieved using the Object Spy.

**Options**

Option /d (for debug) can be used for troubleshooting purposes. When this option is set the whole HTML document of the embedded content is exported into a "dumps" sub-folder of the execution report.

For instance, at:

- C:\Users\<user>\AppData\Local\Temp\SAP\CBTA\Logs\<test script name>\dumps

This folder will contain several text files (one per HTML frames). One may analyze their content to get a better understanding of the HTML content being displayed.

**Explanations**

For more details, refer to: [Support of Embedded HTML Content](#).
**CBTA_GUI_M_Select**

Technical Name: Controls\GuiMenu\Select

The “Select” component selects a menu.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted GuiMenu control. For more information, refer to the URI syntax on page 16.

**CBTA_GUI_PF_SetSecureText**

Technical Name: Controls\GuiPasswordField\SetSecureText

The “SetSecureTest” component sets the value of a secure field.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**THEVALUE**

Specifies the encrypted value of the password.

**OPTIONS**

Reserved for future use

**Note**

- This component supports password encryption using the QTP Password Encoder.
- The SapPassword used by the LaunchAndLogin component relies on the same encryption algorithm.
CBTA_GUI_RB_GetSelected

Technical Name: Controls\GuiRadioButton\GetSelected

The “GetSelected” component gets the state of the target radio button element.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

EXPECTEDVALUE
Specifies the expected state (True or False).

TARGETFIELD
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

Component Output

OUTPUT
This component has an output parameter which receives the radio button state. The subsequent components can use its value as input parameters.

Note
This component has standard behavior. See section Getter Components for more details.
**CBTA_GUI_RB_SelectRadioButton**

Technical Name: Controls\GuiRadioButton\SelectRadioButton

The "SelectRadioButton" component selects a radio button in a group.

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**THEPOSITION**
Specifies the position of the element to select in the radio button group (starting at 1).

**CBTA_GUI_RB_SetSelected**

Technical Name: Controls\GuiRadioButton\SetSelected

The "SetSelected" component selects a radio button.

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**THEVALUE**
Specifies the new state (true or false).

**Note**
No operation is performed if the specified value is empty.
CBTA_GUI_SB_GetMessageParam

Technical Name: Controls\GuiStatusbar\GetMessageParameter

The “GetMessageParameter” component gets the value of one of the message parameters of the Status Bar.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the target status bar. For more information, refer to the URI syntax on page 16.

**INDEX**

Specifies the index of the parameter (starting at 0).

**TARGETFIELD**

This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**

This component has an output parameter which receives the message parameter. The subsequent components can use its value as input parameters.

CBTA_GUI_SB_GetMessageParams

Technical Name: Controls\GuiStatusbar\GetMessageParameters

The “GetMessageParameters” component gets the values of all message parameters in the status bar.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the target status bar. For more information, see the URI syntax on page 16.

**EXPECTED_MESSAGE_TYPE**

Specifies the expected message type.

The possible message types are:

- S  Success
- W  Warning
- E  Error
- A  Abort
- I  Information

**OPTIONS**

/s (save) – Puts the message parameter values in the CBTA Execution Context.
Component Output

This component has six output parameters which receive information about the message parameters. The subsequent components can use their values as input parameters.

MESSAGE

Output parameter which receives the actual message type

MESSAGE TEXT

Output parameter which receives the actual message text, as shown to the user

MESSAGE PARAMETER 0

Output parameter which receives the parameter #0 of the message being displayed

MESSAGE PARAMETER 1

Output parameter which receives the parameter #1 of the message being displayed

MESSAGE PARAMETER 2

Output parameter which receives the parameter #2 of the message being displayed

MESSAGE PARAMETER 3

Output parameter which receives the parameter #3 of the message being displayed

Note

When the ExpectedMessageType parameter is specified, the runtime library checks whether the retrieved value matches the expected one.

- If not, the test reports an error.
- Leave the parameter empty, to avoid checking the value

With the /s option, all message parameters are put in the CBTA execution context, to make the information available to subsequent components. This creates a dedicated variable per message parameter. The name of the variable is prefixed by the current transaction code.

The subsequent component can retrieve the persisted values using the corresponding tokens:

- %VA21_MessageStatus%
- %VA21_MessageParameter0%
- %VA21_MessageParameter1%

Miscellaneous

The Process Flow Analyzer uses this component to persist the result of the last message of type “Success” (S). This is helpful for testing complex business scenarios, in which the result of the first transaction is an input parameter of the next one.
**CBTA_GUI_SB_GetMessageType**

Technical Name: Controls\GuiStatusbar\GetMessageType

This component gets the value of one of the `MessageType` properties in the status bar.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the target status bar. For more information, refer to the URI syntax on page 16.

**EXPECTED VALUE**

Specifies the expected message type.

The possible message types are:

- S Success
- W Warning
- E Error
- A Abort
- I Information

**TARGET FIELD**

This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**

This component has an output parameter which receives the message type. The subsequent components can use its value as input parameters.

**Miscellaneous**

The Process Flow Analyzer uses this component to ensure that the behavior of a test scenario is consistent at runtime. The runtime library assumes that the message type recorded by a PFA matches the message type that the component retrieves during the test. This check is made even if the initial message type was an error (E).
**CBTA_GUI_T_SelectTab**

Technical Name: Controls\GuiTab\Select

The “Select” component selects a tab in a GuiTabstrip.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the [URI](#) syntax on page 16.

---

**CBTA_GUI_TS_GetSelectedTab**

Technical Name: Controls\GuiTabStrip\GetSelectedTab

The “GetSelectedTab” component gets the index of the currently selected tab page within a GuiTabStrip.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the GuiTabStrip container. For more information, refer to the [URI](#) syntax on page 16.

**EXPECTED VALUE**

Specifies the expected index. The test fails, and reports an error, if the selected tab is not the expected one.

**TARGET FIELD**

This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to the [Getter Components](#) section.

**Component Output**

**OUTPUT**

This component has an output parameter which receives the selected tab. The subsequent components can use its value as input parameters.

**Note**

This component has standard behavior. See section [Getter Components](#) for more details.
**CBTA_GUI_TC_GetCellData**

Technical Name: Controls\GuiTableControl\GetCellData

The “GetCellData” component gets the value of a cell in a table.

This operation can be performed using the generic Controls\GetText component, which is faster.

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**ROW**
Specifies the row number (starting at 0).

**COLUMN**
Specifies the column number (starting at 0).

**TARGETFIELD**
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**
This component has an output parameter which receives the cell value. The subsequent components can use its value as input parameters.
CBTA_GUI_TC_SetCellData

Technical Name: Controls\GuiTableControl\SetCellData

The “SetCellData” component sets the value of a cell in a table.

This operation can be performed using the Controls\SetText generic component, which is faster.

Component Parameters

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**ROW**

Specifies the row number (starting at 0).

**COLUMN**

Specifies the column number (starting at 0) or the column title.

**THEVALUE**

Specifies the new value of the cell.
**CBTA_GUI_TC_IsRowSelected**

Technical Name: Controls\GuiTableControl\IsRowSelected

The "IsRowSelected" component determines whether the specified row is part of the current selection.

**Component Parameters**

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**INDEX**
Specifies the row number (starting at 0).

**TARGETFIELD**
This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**
This component has an output parameter which receives the row state. The subsequent components can use its value as input parameters.
CBTA_GUI_TC_FindRow

Technical Name: Controls\GuiTableControl\FindRowByContent

This component searches for a row by content in a GuiTableControl.

Component Parameters

URI
Specifies the uniform resource identifier of the parent table (of type GuiTableControl). For more information, refer to the URI syntax on page 16.

COLUMN TITLE
Specifies the title of the column, which is visible to the end user.

OPERATOR
The operator is a Boolean operator to compare the actual value with the expected one. Refer to the section “Checkpoint Operators” for more details.

CELL CONTENT
Specifies the value to search for.

OPTIONS
Refer to the section “Checkpoint Options” for more details.

Options to Triggering Actions

The options below define the action to perform once the row has been found.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Select</td>
<td>When this option is specified the first row matching the criteria is selected. This to avoid having to select the row using the CBTA_WEB_SELECTROW component.</td>
</tr>
<tr>
<td>/Scroll</td>
<td>This option is implicit (and cannot be removed)</td>
</tr>
</tbody>
</table>

OUTPUT

This component has an output parameter which receives the row number.

- The %Output% and %Row% tokens provide the same information. (new 3.0.8)
- The %Index% token provides the row index (new 3.0.8)
CBTA_GUI_TC_SelectRow

Technical Name: Controls\GuiTableControl\SelectRow

The “SelectRow” component selects or deselects the specified row.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

INDEX
Specifies the row number (starting at 0).

THE_VALUE
Specifies whether to select or unselect the row (True or False).

CBTA_GUI_TXTE_DoubleClick

Technical Name: Controls\ GuiTextEdit\ DoubleClick

The “DoubleClick” component emulates a mouse double click on a GuiTextEdit object.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

CBTA_GUI_TXTE_Press_F4

Technical Name: Controls\ GuiTextEdit\ PressF4

The “PressF4” component emulates pressing the F4 key on a GuiTextEdit object.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.
**CBTA_GUI_TB_PressButton**

Technical Name: Controls\GuiToolbarControl\PressButton

The "PressButton" component emulates pressing a button in the target toolbar control.

**Component Parameters**

**Uri**

Specifies the uniform resource identifier of the targeted GuiToolbarControl object. For more information, refer to the **Uri** syntax on page 16.

**Id**

ID of the button to be pressed (not the ID of the toolbar).

**CBTA_GUI_TB_PressCtxtButton**

Technical Name: Controls\GuiToolbarControl\PressContextButton

This component emulates pressing a context button in the target toolbar control.

**Component Parameters**

**Uri**

Specifies the uniform resource identifier of the targeted GuiToolbarControl object. For more information, refer to the **Uri** syntax on page 16.

**Id**

ID of the button to be pressed (not the ID of the toolbar).

**Example**

This example shows how to press a context button within a toolbar.

**PARAMETERS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uri</td>
<td>label=shell[0]; type=GuiShell; id=/app/con[0]/ses[0]/wnd[0]/shellcont/shell/shellcont[1]/shell[0]</td>
</tr>
<tr>
<td>Id</td>
<td>SELECT</td>
</tr>
</tbody>
</table>
CBTA_GUI_TB_SelectMenuItem

Technical Name: Controls\GuiToolbarControl\SelectContextMenuItem

This component selects an item from the context menu of the button targeted in a toolbar control. The component for opening the menu does not exist; this operation is implicit at runtime when executing a test.

Component Parameters

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**ID**

Specifies the ID of the button.

**FUNCTIONCODE**

Specifies the function code of the item to be selected

By default, the component expects the code of the context menu item. The text can be specified instead, by setting the Options parameter to /pt.

**OPTIONS**

- /pt – Use this option when the FunctionCode parameter provides the text, instead of the code of the context menu item.
CBTA_GUI_T_ChangeCheckbox

Technical Name: Controls\GuiTree\ChangeCheckbox

This component changes the state of a checkbox in a tree control.

Component Parameters

URI

"URI" specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

NODEID

The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

COLUMNNAME

The name of the column

THESTATE

The new state of the targeted checkbox

OPTIONS

The “Options” parameter can declare whether the specified NodeId corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>/k (key)</th>
<th>Use this option when the NodeKey is specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pt</td>
<td>This option can search for the node using its text.</td>
</tr>
</tbody>
</table>
CBTA_GUI_T_ClickLink

Technical Name: Controls\GuiTree\ClickLink

The component emulates a mouse click on a link in a tree control.

Component Parameters

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**NODEID**

Specifies the NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**COLUMNNAME**

Specifies the name of the column.

**OPTIONS**

The “Options” parameter can be used to declare whether the specified NodeId corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k (key)</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option can search for the node using its text.</td>
</tr>
</tbody>
</table>
**CBTA_GUI_T_CollapseNode**

Technical Name: Controls\GuiTree\CollapseNode

The component collapses a tree node.

**Component Parameters**

**Uri**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**NODEID**

Specifies the NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the /k option is used.

**OPTIONS**

This parameter can be used to declare whether the NODEID specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k (key)</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

**CBTA_GUI_T_DoubleClickItem**

Technical Name: Controls\GuiTree\DoubleClickItem

This component emulates a mouse double-click on an item in a tree control.

**Component Parameters**

**Uri**

The uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**NODEID**

Specifies the NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**COLUMNNAME**

Specifies the name of the column.

**OPTIONS**

The “Options” parameter can declare whether the specified NODEID corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k (key)</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

**CBTA_GUI_T_DoubleClickNode**

Technical Name: Controls\GuiTree\DoubleClickNode

The component emulates a mouse double-click on a tree node.
Component Parameters

Uri
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

NodeID
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

Options
The “Options” parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

CBTA_GUI_T_ExpandNode

Technical Name: Controls\GuiTree\ExpandNode

The component expands a tree node.

Component Parameters

Uri
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

NodeID
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

Options
The “Options” parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>
CBTA_GUI_T_GetCheckBoxState

Technical Name: Controls\GuiTree\GetCheckBoxState

The component gets the state of a checkbox in a tree control.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**NODEID**
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**COLUMN NAME**
Specifies the name of the column.

**EXPECTEDSTATE**
Specifies the expected state.

**OPTIONS**
The “Options” parameter declares whether the Nodeid specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

Component Output

**OUTPUT**
This component has an output parameter which receives the checkbox state. The subsequent components can use its value as input parameters.
CBTA_GUI_T_PressButton

Technical Name: Controls\GuiTree\PressButton

The "PressButton" component emulates pressing a button in a tree control.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

NODEID
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

COLUMNNAME
Specifies the name of the column.

OPTIONS
The "Options" parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k (key)</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

CBTA_GUI_T_PressHeader

Technical Name: Controls\GuiTree\PressHeader

The component emulates pressing a button in a tree control.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

HEADERNAME
Specifies the name of the header to select.
CBTA_GUI_T_SelectColumn

Technical Name: Controls\GuiTree\SelectColumn

The “SelectColumn” component selects a column of a tree control.

Component Parameters

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**COLUMN NAME**

Specifies the name of the column.

**OPTIONS**

The parameter declares whether the **NodeId** specified corresponds to the **NodeKey** or to the **NodePath**.

| /k (key) | Use this option when the **NodeKey** is specified |
| /pt    | This option searches for the node using its text. |
CBTA_GUI_T_SelectColMenuItem

Technical Name: Controls\GuiTree\SelectColumnContextMenu

The component triggers the selection of an item of the context menu of a certain node and column of the tree. The component for opening the menu does not exist; this operation is implicit at runtime when executing a test.

Screenshot of Transaction KEPM – Planning Framework

Figure 14: CBTA_GUI_T_SelectColMenuItem Example

Component Parameters

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**NODEID**

Specifies the NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the /k option is used.
**COLUMNNAME**
Specifies the technical name of the column.

**FUNCTIONCODE**
Specifies the menu item to be selected.

**OPTIONS**
The “Options” parameter can be used to declare whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/k (key)</td>
<td>Use this option when the NodeKey is specified</td>
</tr>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>
**CBTA_GUI_T_SelectMenuItem**

Technical Name: Controls\GuiTree\SelectContextMenuItem

The component emulates selecting an item in the context menu of the target tree. The component for opening the menu does not exist; this operation is implicit at runtime when executing a test.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**NODEID**

The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**FUNCTIONCODE**

The menu item to be selected.

**OPTIONS**

The "Options" parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>/k (key)</th>
<th>Use this option when the NodeKey is specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>
CBTA_GUI_T_SelectItem

Technical Name: Controls\GuiTree\SelectItem

The "SelectItem" component selects an item in a tree control.

Component Parameters

**Uri**
The uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

**NodeId**
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**COLUMN NAME**
The name of the column

**OPTIONS**
The "Options" parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>/k (key)</th>
<th>Use this option when the NodeKey is specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

**Example**

This example shows how to select an item within a tree.

**PARAMETERS**

<table>
<thead>
<tr>
<th>Uri</th>
<th>label=Tree_SAP.TableTreeControl.1; type=GuiTree; id=/app/con[1]/ses[0]/wnd[0]/usr/cntICCONTAINER/shellcont/shell/shellcont[1]/shell[1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodeld</td>
<td>1</td>
</tr>
<tr>
<td>Column Name</td>
<td>C 1</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>

**SCREENSHOT OF TRANSACTION MMBE – STOCK OVERVIEW**

![Screenshot of Transaction MMBE - Stock Overview](image)

Figure 15: CBTA_GUI_T_SelectItem Example
CBTA_GUI_T_SelectNode

Technical Name: Controls\GuiTree\SelectNode

The "SelectNode" component selects the specified node in a tree control.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**NODEID**
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**OPTIONS**
The "Options" parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>/k (key)</th>
<th>Use this option when the NodeKey is specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

CBTA_GUI_T_SetCheckBoxState

Technical Name: Controls\GuiTree\SetCheckBoxState

The "SetCheckBoxState" component changes the state of a checkbox in a tree.

Component Parameters

**URI**
Specifies the uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**NODEID**
The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option /k is used.

**COLUMN NAME**
Specifies the name of the column.

**THESTATE**
Specifies the new state of the checkbox.

**OPTIONS**
The "Options" parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>/k (key)</th>
<th>Use this option when the NodeKey is specified</th>
</tr>
</thead>
</table>
CBTA_GUI_T_UnselectAll

Technical Name: Controls\GuiTree\UnselectAll

The “UnselectAll” component resets all selections in the target tree control.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

OPTIONS
Reserved for future use

CBTA_GUI_T_UnselectColumn

Technical Name: Controls\GuiTree\UnselectColumn

The “UnselectColumn” component deselects a column of a tree control.

Component Parameters

URI
Specifies the uniform resource identifier of the targeted object. For more information, refer to the URI syntax on page 16.

COLUMNNAME
Specifies the name of the column.

OPTIONS
Reserved for future use

Example

This example shows how to deselect a column within a tree.

PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uri</td>
<td>label=Tree_SAP.TableTreeControl.1; type=GuiTree; id=/app/con[1]/ses[0]/wnd[0]/usr/cntlCC_CONTAINER/shellcont/shell/shellcont[1]/shell[1]</td>
</tr>
<tr>
<td>ColumnName</td>
<td>C 1</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>

In this example, the Column Name includes space characters. Specify the exact value as you retrieve it using the CBTA Object Spy.
**CBTA_GUI_T_UnselectNode**

Technical Name: Controls\GuiTree\UnselectNode

The component unselects a tree node.

**Component Parameters**

**Uri**

The uniform resource identifier of the targeted object. For more information, refer to the **URI** syntax on page 16.

**NodeId**

The NodePath or the NodeKey of the targeted node. By default, the component expects a node path, unless the option `/k` is used.

**Options**

The options parameter declares whether the NodeId specified corresponds to the NodeKey or to the NodePath.

<table>
<thead>
<tr>
<th>/k (key)</th>
<th>Use this option when the NodeKey is specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pt</td>
<td>This option searches for the node using its text.</td>
</tr>
</tbody>
</table>

**Example**

This example shows how to deselect a node within a tree.

**PARAMETERS**

<table>
<thead>
<tr>
<th>Uri</th>
<th>label=shell; type=GuiShell; id=/app/con[0]/ses[0]/wnd[0]/usr/cntlIMAGE_CONTAINER/shellcont/shell /shellcont[0]/shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeId</td>
<td>2</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
</tbody>
</table>
6.5 SAP GUI - Test Automation Challenges

CBTA_GUI_T_SelectColMenuItem – How to Use It

This component selects an item in the context menu of a node of the tree, and a column. This component is complex and determining the values of its parameters can be difficult.

**EXAMPLE OF CONTEXT MENU ITEM**

![Planning Framework: Overview](image)

Figure 17: CBTA_GUI_T_SelectColMenuItem Example

**What are the component parameters?**

The component expects the following parameters:

- Container, the tree URI
- ID, the node in the tree
- Technical name of the column
- Function code identifying the item to be selected

None of the parameters require language-dependent values, so tests using this component can run with different language settings.
How to retrieve the parameter value?

RECORDING
Determine the value of the input parameter by recording the scenario using the CBTA. The generated test is automatically populated with the relevant values, and the test works.

SPYING USING THE OBJECT SPY
Determining the value without recording is difficult because some dynamic parts of the UI such as the context menu are not visible when using the CBTA Object Spy.

The screenshot below shows the information available when spying the node in the tree. It shows:

- The technical name of the columns of the tree
- The technical information of the node, such as the node path
- The URI of the tree

![Figure 18: Object Spy - GuiTree Information](image)

The Object Spy cannot display context menu information, so you cannot determine the “FunctionCode” here. Use the native SAP GUI recording capabilities.
**RECORDING USING SAP FRONT END RECORDER**

The SAP front end and the underlying SAP GUI technology can record user interactions. The recording can be started directly, as shown in the screenshot below.

A dedicated toolbar lets you record a scenario or play back an existing one.

To start the recording, press the red icon. Select the item in the context menu, and then stop the recording.

The recorded scenario is persisted on the file system in a VB script format.

The generated VB script can be opened using any text editor. The parameters required to perform the user action are visible like regular VB script parameters.

In our example, the expected value of the **FunctionCode** parameter is “PLPCK_CH01”.

![Figure 19: SAP GUI Scripting Recorder](image-url)
For convenience, the “CBTA_GUI_T_SelectColMenuItem” component performs two atomic operations at once. It emulates the right-click on the item (which is visible here as a call to the “itemContextMenu” function), and selects the item.

The NodeKey or the NodePath can be specified as NodeId. By default, the CBTA recording generates a test script using the NodePath. The /k (for key) option passes the NodeKey instead of the NodePath, to identify the targeted node in the tree.

```
If Not IsObject(application) Then
    Set SapGuiAuto = GetObject("SAPGUI")
    Set application = SapGuiAuto.GetScriptingEngine
End If
If Not IsObject(connection) Then
    Set connection = application.Children(0)
End If
If Not IsObject(session) Then
    Set session = connection.Children(0)
End If
If IsObject(MScript) Then
    WScript.ConnectObject session, "on"
    WScript.ConnectObject application, "on"
End If
session.findById("wnd[0]").resizeWorkingPane 91,13,False
session.findById("wnd[0]/shellcont/shell/shellcont[0]/shell"
    .selectContextMenuItem "FLPCR_CH01"
```
Dynamic SAP GUI Scenarios

Automating business scenario tests can be complex if the behavior of the application differs depending on the external information from the database. For such dynamic scenarios, the GUI controls cannot be found using their ID, because it changes each time the test is performed.

CBTA addresses this issue by allowing you to:

- Search for controls using their text instead of their ID.
- Search for child elements using their text instead of their node path or node key.
- Customize the runtime library with customer-specific subroutines and functions.

Searching for SAP GUI Controls by Text

Default components targeting GUI controls expect a URI as first input parameter. The URI syntax is flexible and is normally used to search for controls using their ID or name. You can also search by text when testing a SAP GUI application.

URI Syntax with Text:

The URI is composed of key-value pairs, separated by a semicolon and a space character. When searching by text, the URI must have the format:

```
type=<controlType>; text=<controlText>
```

An additional index can be specified, to avoid ambiguities when several controls have the text.

```
type=<controlType>; text=<controlText>; index=<controlIndex>
```

Refer to page 16 for more information about the URI syntax.
Searching for Child Elements by Text

Complex SAP GUI controls such as GuiTree, GuiToolBarControl and GuiGridView, are containers. They include a set of child elements with which the user can interact.

The tester may need to find a child element using its text instead of its ID. The URI syntax cannot do this, because the URI only identifies the parent control (the container), not its child element.

Some default components that target child elements have an "Options" parameter.

- This parameter must be set to /t when searching the child element by text.
- This parameter must be set to /pt when a path is specified (several texts separated by a backslash). This is typically used when searching for a node by text in a tree.

Example with Component: GuiTree/SelectItem

This example shows how to select a child node in the GuiTree using the text of each node (instead of the node path).

PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>label=Tree_SAP.TableTreeControl.1; type=GuiTree; id=/app/con[1]/ses[0]/wnd[0]/usr/cntlCC_CONTAINER/shellcont/shell/shellcont[1]/shell[1]</td>
</tr>
<tr>
<td>NodeId</td>
<td>nodeText1\nodeText2\nodeText3</td>
</tr>
<tr>
<td>ColumnName</td>
<td>C 1</td>
</tr>
<tr>
<td>Options</td>
<td>/pt</td>
</tr>
</tbody>
</table>
DoFileUpload Custom Function

Uploading a file cannot be recorded because some user actions are made via a native File Upload dialog. The alternative is to insert a step calling a DoFileUpload custom function.

Example of a Native File Upload dialog

![Figure 21: Internet Explorer - File Upload Dialog](image)

Customizing the Runtime Library

The prerequisite is, of course, to create the DoFileUpload custom function. This can be done using the Runtime Library Manager and its Code Assistant.

![Figure 22: Runtime Library Manager - Code Assistant](image)
Selecting a Code Pattern

The `DoFileUpload` function can be created in a few clicks by selecting the corresponding pattern that SAP delivers.

![Runtime Library - Code Pattern Selection](image)

Figure 23: Runtime Library - Code Pattern Selection

Adaptation of the Recorded Test Script

As already mentioned, the test script must be adapted manually. The component `CBTA_WEB_InvokeFunction` must be inserted just before the step responsible for opening the `File Upload` dialog like shown in this example:

![Calling the DoFileUpload custom function](image)

Figure 24: Calling the DoFileUpload custom function

Note that in this example the step that the recorder generated for selecting the menu item has been disabled on purpose - our `DoFileUpload` function performs this action as well.

Calling the Custom Function

The name of the library and the name of our custom function must be specified.

![DoFileUpload Input Parameters](image)

Figure 25: DoFileUpload Input Parameters
Input Parameters

- Parameter1 – URI of the button opening the File Upload dialog
- Parameter2 – FileName – full name of the file that is to be uploaded
- Parameter3 – Not used
- Parameter4 – Not used
- Options
  - /menu - URI refers to a MENU ITEM instead of a button
Embedded HTML Content

SAP GUI transactions have the capacity to embed some HTML content using the GuiHTMLViewer control.

The test automation of the embedded content is challenging because:

- Actions made against the HTML content embedded in SAP GUI Transactions are not recorded by default.
- A manual adaptation of the generated test script might be necessary.
- At runtime, the CBTA_GUI_HV_StartWebController component must be used to attach the test player to the embedded Internet Explorer session.
- Starting with CBTA 3.0 SP05 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.

Note

For more information, see the documentation:

CBTA – Object Spy – Troubleshooting Tool
SAP GUI Scripting API

When testing SAP GUI transactions, the CBTA components perform actions using the SAP GUI Scripting API.

This API is well documented, and after client setup its help file is usually available on the file system at:

- C:\Program Files (x86)\SAP\FrontEnd\SAPgui\SAPguihelp\SAPGUIScripting.chm

You may use it to find the properties that are exposed by SAP GUI Scripting objects.

Here is an example for the GuiTextField type:

![Figure 27: SAP GUI Scripting API](image)
7 SAP CRM / WebCUIF

CBTA delivers components to automate the testing of CRM Web applications that are built on top of the WebCUIF Layer. Since SAP CRM is the main product using this WebCUIF UI Framework, all components supporting this technology are prefixed using "CBTA_CRM_".

7.1 URI Identifying CRM UI Elements

Like the SAP GUI components, the SAP CRM components that target a UI element have at least a URI parameter. The URI syntax has been extended to support SAP CRM needs and its complex web page composition.

Usually, the test engineer cannot determine the URI value on his own. The recommendation is to create the test script by recording the business scenario, and then use the Object Spy to solve execution issues.

URI Syntax for SAP CRM

The URI is composed of name-value pairs, separated by a semicolon and a trailing space character. For example:

```
label=<uiElementLabel>; tag=<tagName>; crm.id=<uiElementId>
```

A URI that targets a CRM UI element provides at least the crm.id attribute.

- The crm.id is used to search the frames for the UI element.
- The tag attribute can be used to specify the HTML element type that the test targets. This is used when the internal HTML structure is complex – when several HTML elements are combined together to render a CRM control (such a dropDownListbox).
- Additional information is not mandatory. A label can be specified as well. However, this information is not used to search for the target. This information is simply used as a comment and shown in the execution report.

Examples of valid URI:

```
label=Title; tag=INPUT; crm.id=A_contactV_detailsT_inputFieldC_headerI_struct.title_key
```

The crm.id value generated by the CRM WebCUIF framework contains meta-information about the data being displayed.

This example provides the following meta-information:

- Application: contact
- View Name: details
- Type: inputField
- Context Name: header
- Interface: struct.title_key

To make sure it generates stable tests, CBTA uses the information as it is. The CRM WebCUIF framework guarantees that this meta-information remains the same for all browser and user sessions.
Targeting a different HTML frame

The HTML content generated by the CRM WebCUIF framework consists of several HTML documents, that are nested by HTML frames (using the FRAME or IFRAME tag). This can lead to naming conflicts when performing actions against HTML elements. To avoid these conflicts and to make it easier to test CRM applications, the SAP CRM components assume (by default) that the target of the action is located in the work area frame.

If the target is in another frame, an additional *crm.area* attribute is necessary.

**Examples of URI specifying the frame:**

URI targeting the “Log Off” button in the *Header* frame

```
label=Log Off; tag=A; id=LOGOFF; crm.area=HeaderFrame
```
7.2 SAP CRM - Action Components

**CBTA_CRM_A_CaptureScreen**

Technical Name: Actions\CaptureScreen

This component captures a screenshot of the HTML page as it is displayed by the browser.

The persisted screenshot will be in the generated CBTA report.

**Component Parameters**

**URI**

- Specifies the uniform resource identifier of the targeted frame. If empty, the component captures the complete HTML page.

**OPTIONS**

- Reserved for future use.

**CBTA_CRM_A_GetLastMsgParams**

Technical Name: Actions\GetLastMessageParameters

(Deprecated) This component gets information about the application output.

This component has been deprecated – use the Actions/GetMessageParameters instead.

The information collected is stored in the execution context, to make it available to subsequent components.

Options can also check the behavior of the application and make sure it reports consistent messages to the user.

**Component Parameters**

**MESSAGESOURCE**

- the expected message source.

**MESSAGETYPE**

- the expected message type.

Existing types are:

- “S” for Success
- “I” for Info
- “W” for Warning
- “E” for Error

**MESSAGEID**

- the expected message ID.
MESSAGE NUMBER

- the expected Message Number.

OPTIONS

- /c (check) – checks that the message type is the expected one. Only the message type is checked. Other parameters are not used in this case.
- /a (all) – checks all parameters (source, type, ID and number) and reports test failure if one of them differs from the expectation.

CBTA_CRM_A_GetMessageParams

Technical Name: Actions\GetMessageParameters

This component gets information about the application output. The information collected is stored in the execution context to make it available to subsequent components.

Options can also check the behavior of the application and make sure it reports consistent messages to the user.

Component Parameters

MESSAGE ID

- the expected message ID.

MESSAGE NUMBER

- the expected message number.

EXPECTED MESSAGE TYPE

- the expected message type.

Existing types are:

- "S" for Success
- "I" for Info
- "W" for Warning
- "E" for Error

Component Output

The component searches the message container for the first message matching the specified MessageId and the MessageNumber. It reports an error if the message is not found.

If the ExpectedMessageType parameter is specified, the component also checks whether the actual message type is the expected one.

If the message is found, the message parameters are collected, stored in the execution context, and returned as output parameters. The following output parameters are populated:

- MessageType – The actual message type
- MessageText – The complete text being displayed
- MessageParameter1 – The parameter #1 of this message
- MessageParameter1 – The parameter #2 of this message
- MessageParameter1 – The parameter #3 of this message
- MessageParameter1 – The parameter #4 of this message

**CBTA_CRM_A_LaunchAndLogin**

Technical Name: Actions\LaunchAndLogin

This component initializes the SAP CRM session. All the subsequent components use this session by default.

**Component Parameters**

**SAP_SYSTEM**
- the URL to start in the browser.

**SAP_SYSTEM**
- the SAP system of the CRM application.

**SAP_CLIENT**
- the client number.

**SAP_USER**
- the name of the user to logon to the system.

**SAP_PASSWORD**
- the password of the user.

**SAP_LANGUAGE**
- the preferred language - example: EN (for English) or JA (for Japanese).

**SAP_ROLE**
- the role of the user - e.g.: SALESPRO.

**SAP_LOGICAL_LINK**
- the logical link to the CRM component to start.
CBTA_CRM_A_LogOff

Technical Name: Actions\LogOff

This component logs the user off the SAP CRM system.

Component Parameters

URI

- Specifies the URI of the button which triggers the user log off. Leave it empty to get the default behavior. The component confirms the operation automatically.

Figure 28: CBTA_CRM_A_LogOff Example
**CBTA_CRM_A_ClosePopup**

Technical Name: Actions\Popup\ClosePopup

This component emulates a click in the native cross icon (part of the window title) which closes a modal popup window. Closing a popup is the action that the tester performs implicitly when an action targets the main browser window, so this component is optional and not used when generating tests using the PFA.

**Component Parameters**

**URI**

Specified the uniform resource identifier of the targeted popup.

For instance:

- Set the URI parameter to "popupId=1" to close the modal popup #1
- Set the URI parameter to "popupId=2" to a modal popup #2 (a modal popup on another modal popup)

**OPTIONS**

- Reserved for future use

![CBTA_CRM_A_ClosePopup Example](image-url)
7.3 SAP CRM - Generic Components

**CBTA_CRM_CheckAttribute**

Technical Name: Controls\CheckAttribute

This component checks attributes selected by the Check Picker when recording scenarios.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted HTML element.

**ATTRIBUTENAME**
- the name of the HTML attribute to be checked.

**OPERATOR**
- the Boolean operator to compare the actual and expected values. See the “Checkpoint Operators” section for more details.

**EXPECTEDVALUE**
- the expected value.

**OPTIONS**
- The options parameter converts the type before the comparison. See the “Checkpoint Options” section for more details.

**Component Output**

**OUTPUT**
- This component has an output parameter which receives the value of the attribute. The subsequent components can use its value as input parameters.
CBTA_CRM_CheckProperty

Technical Name: Controls\CheckProperty

This component retrieves and checks the value exposed by UI elements via HTML attributes. It checks properties selected by the Check Picker when recording scenarios.

Component Parameters

**URI**
- the uniform resource identifier of the targeted HTML element.

**PropertyName**
- the name of the HTML property from which to retrieve the value.

For HTML elements displaying a text, the *innerText* property provides the displayed information. For HTML elements such as input fields, the *innerText* property is not relevant. The value attribute provides the information and the CheckAttribute component checks.

If the property name is “exist” it checks whether the UI element (identified by the URI) exists in the HTML content.

**Operator**
- the Boolean operator which compares the actual and expected values. See the “Checkpoint Operators” section for more details.

**ExpectedValue**
- the expected value. For boolean properties (such as “exist”) the expected value must be True or False.

**Options**
- The options parameter converts the type before the comparison. See the “Checkpoint Options” section for more details.

Component Output

**Output**
- This component has an output parameter which receives the value of the property. The subsequent components can use its value as input parameters.
**CBTA_CRM_Click**

Technical Name: Controls\Click

This component emulates a click on an HTML element.

*Component Parameters*

**Uri**

- the uniform resource identifier of the targeted HTML element.

**CBTA_CRM_GetAttribute**

Technical Name: Controls\GetAttribute

This component retrieves the value exposed by UI elements via HTML attributes.

*Component Parameters*

**Uri**

- the uniform resource identifier of the targeted HTML element.

**AttributeName**

- the name of the HTML attribute from which to retrieve the value.

**ExpectedValue**

- the expected value.

**TargetField**

- This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

*Component Output*

**Output**

- This component has an output parameter which receives the value of the attribute. The subsequent components can use its value as input parameters.
**CBTA_CRM_GetProperty**

Technical Name: Controls\GetProperty

This component retrieves the value exposed by UI elements via their HTML properties.

**Component Parameters**

**Uri**
- the uniform resource identifier of the targeted HTML element.

**PropertyName**
- the name of the HTML property from which to retrieve the value.

**ExpectedValue**
- the expected value.

**TargetField**
- This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to [Getter Components](#) section.

**Component Output**

**Output**
- This component has an output parameter which receives the value of the property. The subsequent components can use its value as input parameters.
**CBTA_CRM_PressKey**

Technical Name: Controls\PressKey

This component emulates a keystroke targeting an HTML element.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted HTML element.

**KEYCODE**
- the code identifying the key to press.

**Explanations**

The code is a string identification key, for instance: ENTER, F3, UP, DOWN. Keystroke combination is possible - e.g.: CTRL+ENTER, SHIFT+F3 or ALT+S.

Most common keyboard actions, such as those for navigation (e.g. TAB) are not useful in tests.

**CBTA_CRM_SetAttribute**

Technical Name: Controls\SetAttribute

This component sets the value of an HTML attribute of a UI element.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted HTML element.

**ATTRIBUTENAME**
- the name of the HTML attribute to modify.

**THEVALUE**
- the new value of the attribute.

**CBTA_CRM_SetProperty**

Technical Name: Controls\SetProperty

This component set the value of an HTML property of a UI element.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted HTML element.

**PROPERTYNAME**
- the name of the HTML property to modify.
**TheValue**

- the new value of the property.

---

**CBTA_CRM_SetFocus**

Technical Name: Controls\SetFocus

This component puts the focus on an HTML element.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted HTML element.
7.4 SAP CRM - Control Components

CBTA_CRM_BTN_ClickButton

Technical Name: Controls\Button\ClickButton

This component emulates a click on a button.

Component Parameters

URI

- the uniform resource identifier of the targeted HTML element.

CBTA_CRM_BTN_SetButtonState

Technical Name: Controls\Button\SetButtonState

This component changes the state of a collapsible button. The operation is only performed when the current state differs from the expected one.

Component Parameters

_URI

- the uniform resource identifier of the targeted HTML element.

_STATE

- the new state – expected values are either “expanded” or “collapsed”.

Examples

Examples with two collapsible buttons:

Figure 30: SAP CRM - Collapsible Buttons
**CBTA_CRM_CB_GetSelected**

Technical Name: Controls\Checkbox\GetSelected

This component retrieves whether a checkbox UI element is selected.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted HTML element.

**EXPECTED VALUE**

- the expected value, True or False.

**TARGET FIELD**

- This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**

- This component has an output parameter which receives the checkbox state. The subsequent components can use its value as input parameters.

**CBTA_CRM_CB_SetSelected**

Technical Name: Controls\Checkbox\SetSelected

This component modifies the state of a checkbox UI element.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted HTML element.

**THE VALUE**

- the new state of the checkbox, True or False.

**CBTA_CRM_DP_OpenDatePicker**

Technical Name: Controls\DatePicker\OpenDatePicker

(Deprecated) This component opens the Date Picker popup, to select a date.

This component is obsolete and never used in tests generated by the PFA. If a date is entered, the recorded value is collected and re-used in a test. When manipulating dates, use tokens like %today% or %tomorrow%, and assign the value directly to the input field without opening the calendar.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted HTML element.
**CBTA_CRM_DP_SELECTDATE**

Technical Name: Controls\DatePicker\SelectDate

(Deprecated) This component selects a date in the Data Picker associated to an HTML element.

This component is obsolete and never used in tests generated by the PFA. If a date is entered, the recorded value is collected and re-used in a test. When manipulating dates, use tokens like %today% or %tomorrow%, and assign the value directly to the input field without opening the calendar.

**Component Parameters**

**URI**

- the uniform resource identifier of the date that is to be selected.

**CBTA_CRM_DLBJ_SELECTITEM**

Technical Name: Controls\DropdownListBox\SelectItem

This component emulates a click on a DropdownListBox item.

**Component Parameters**

**URI**

- the uniform resource identifier of the item that is to be selected.

**CBTA_CRM_DLBJ_SELECTKEY**

Technical Name: Controls\DropdownListBox\SelectKey

This component selects a DropdownListBox item using its key.

**Component Parameters**

**URI**

- the uniform resource identifier of the item to be selected.

**THEKEY**

- the key identifying the item to be selected. The key is internal information that can be retrieved by the CBTA Object Spy.

**CBTA_CRM_DLBJ_SELECTVALUE**

Technical Name: Controls\DropdownListBox\SelectValue

This component searches for the DropdownListBox item matching the value specified and selects it. Note that the CBTA_CRM_IF_SETVALUE component can be used as well.

**Component Parameters**

**URI**

- the uniform resource identifier of the item that is to be selected.
the value of the item to be selected.

**CBTA_CRM_IF_GetValue**

Technical Name: Controls\InputField\GetValue

This component retrieves the value of an input field.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted HTML element.

**EXPECTED VALUE**

- the expected value.

**TARGET FIELD**

- This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**

- This component has an output parameter which receives the value of the input field. The subsequent components can use its value as input parameters.

**Explanations:**

This component is similar to the GetAttribute component, with the difference that the GetValue component determines which HTML attribute provides the information, based on the type of the targeted UI element.

**CBTA_CRM_IF_OpenInputHelp**

Technical Name: Controls\InputField\OpenInputHelp

This component emulates the click on the “Value Help” button, which opens a popup when pressing the F4 key.

**Component Parameters**

**URI**

- Specifies the uniform resource identifier of the targeted HTML input element.

**CBTA_CRM_IF_SetValue**

Technical Name: Controls\InputField\SetValue

This component sets the value of an input field.
Component Parameters

**Uri**
- the uniform resource identifier of the targeted HTML input element.

**TheValue**
- the new value of the input field.

**CBTA_CRM_L_ClickLink**

Technical Name: Controls\Link\ClickLink

This component emulates a click on a link.

Component Parameters

**Uri**
- the uniform resource identifier of the targeted HTML anchor element.

**CBTA_CRM_M_OpenSubMenu**

Technical Name: Controls\Menu\OpenSubMenu

(Deprecated) This component emulates the actions opening the context menu of an HTML element.

This component is implicit when using the Menu/SelectMenuItem component.

Component Parameters

**Uri**
- the uniform resource identifier of the targeted HTML element.

**CBTA_CRM_M_SelectMenuItem**

Technical Name: Controls\Menu\SelectMenuItem

This component emulates the selection of an item in the context menu associated with an HTML element.

Component Parameters

**Uri**
- the uniform resource identifier of the targeted item in the context menu.

**CBTA_CRM_SelectRadioButton**

Technical Name: Controls\RadioButton\SelectRadioButton

This component emulates the actions selecting a radio button.
**Component Parameters**

**Uri**
- the uniform resource identifier of the targeted item in the context menu.

**Thekey**
- the new key of the item to be selected.

**CBTA_CRM_NAVB_ClickNavLink**

Technical Name: Controls\NavigationBar\ClickNavigationLink

This component emulates a click on a link in the navigation panel.

**Component Parameters**

**Uri**
- the uniform resource identifier of the targeted HTML anchor element.

**CBTA_CRM_SR_SelectRow**

Technical Name: Controls\SearchResult\SelectRow

This component emulates a click on a cell triggering the selection of a line in tables displaying a search result.

**Component Parameters**

**Uri**
- the uniform resource identifier of the targeted cell identifying the row in the table.
**CBTA_CRM_T_FindRow**

Technical Name: Controls\Table\FindRowByContent

This component searches for a row by content.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the parent table. For more information, refer to the [URI](#) syntax on page 16.

**COLUMN_TITLE**

Specifies the title of the column visible to the end user.

**OPERATOR**

The operator is a Boolean operator to compare the actual and expected values. Refer to the “Checkpoint Operators” section for more details.

**CELL_CONTENT**

Specifies the value to search for.

**OPTIONS**

The options below define the action to perform once the row has been found.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Select</td>
<td>When this option is specified the first row matching the criteria is selected. This to avoid having to select the row using the CBTA_WEB_SELECTROW component.</td>
</tr>
<tr>
<td>/Quiet</td>
<td>(new 3.0.8) This option can be used to avoid reporting an error (in the execution report) when the row is not found.</td>
</tr>
</tbody>
</table>

You can search for the row in various ways. Refer to the “Checkpoint Options” section for more details.

**Component Output**

**OUTPUT**

This component has an output parameter which receives the row number. The subsequent steps may rely on either the output parameter or the %Output% token to reuse the information.

The %Row% token provides the same information. (new 3.0.8)

**CBTA_CRM_T_SelectRow**

Technical Name: Controls\Table\SelectRow

This component emulates a click on a cell triggering the selection of a line in tables.
Component Parameters

Uri
- the uniform resource identifier of the targeted cell identifying the row in the table.

CBTA_CRM_TS_SelectTab

Technical Name: Controls\TabStrip\SelectTab

This component emulates the selection of a tab.

Component Parameters

Uri
- the uniform resource identifier of the tab to select.
7.5 SAP CRM - Query Components

Query components address issues that the test engineer may face while automating SAP CRM business processes.

**CBTA_CRM_SelectTransactionType**

Technical Name: Queries\SelectTransactionType

Some SAP CRM business processes start by asking the end user to select a “Transaction Type” in a modal popup. This modal popup is a typical example of where the information to be selected must be searched by its text (not using a technical ID).

The screenshot below illustrates this situation. The first column shows the transaction type and the second column a short description. Both columns can identify the transaction type to be selected.

![SAP CRM - Transaction Type Selection](image)

The bottom of the screen includes a "pager" control, which navigates to the next pages when the transaction type is not on the current page.

**Component Parameters**

**TRANSACTIONTYPE**

the transaction type to search for; e.g.: TA as shown in the previous screenshot.

**DESCRIPTION**

the transaction type description to search for. This input parameter is optional, and can be left empty to avoid searching by the description.

**OPERATOR**

the Boolean operation to compare values. The “=” operator is the default.
MaxPage

the number of pages in which to look for the transaction type. Default value is 10.

Options

comparison options.

The option parameter can take the following values:

- /t (trimming) ➔ Ignore leading and trailing spaces when comparing the TransactionType and Description with the values of the respective columns.
- /u (upper-case) ➔ for a case-insensitive comparison
CBTA_CRM_SelectMenuItemByText

Technical Name: Queries\SelectMenuItemByText

Some SAP CRM scenarios generate context menus in which the child items do not have a stable ID. This component finds these items using the text visible to the end user.

Component Parameters

URI
the URI of the CRM control with which the context menu is associated.

OPERATOR
the Boolean operator to check the text of the item.

TEXT
the text of the item to search for.

VIEWNAME
the view name to filter out items that are not in the scope of the current action.

OPTIONS
The option parameter can take the following values:

- /t (trimming) ➔ Ignore leading and trailing spaces when comparing the TransactionType and Description with the values of the respective columns.
- /u (upper-case) ➔ for a case-insensitive comparison

Additional options can influence the behavior of this default component:

- /click ➔ performs a mouse click on the control identified by the URI, to open the context menu before searching for the child menu item.
8 Web Applications

CBTA delivers a set of components to automate testing of web applications.

As already mentioned, several UI technologies are supported:

- **Web** – UI technologies which display their content using HTML tags like:
  - BSP
- **Unified Rendering Light Speed (LS)** – UI layer common to most of the SAP UI frameworks, like:
  - Web Dynpro ABAP
  - Web Dynpro Java - (version based on Light Speed only)
  - Web GUI – a.k.a. SAP GUI for HTML

Most of the components are generic enough to perform the action on any UI technology, so the component to set the value of an input field might be the same for applications based on Web Dynpro ABAP, Web Dynpro JAVA, and others.

There are several component categories. Some components perform general actions, some set the value (or change the state) of a UI element, some retrieve information from the HTML page.
8.1 URI Identifying HTML UI Elements

Like the SAP GUI components, the SAP CRM components that target a UI element have at least a URI parameter. The URI syntax is quite flexible; it has been extended to support different Web UI technologies. It allows searching for UI elements using various strategies.

HTML Elements and Documents

Web pages comprise one or more HTML documents. Each HTML document includes a collection of HTML elements consisting of tags enclosed in angle brackets (like <html>, <body>, or <input>).

- The tag of the HTML element denotes its type.
- HTML elements may have an ID or a name.
- HTML elements may have additional HTML attributes (other than the ID and the name attributes).
- HTML elements have properties (such as the innerText).

Web Controls versus HTML Elements

With modern applications, the HTML content, which the browser displays, is generated on the server by a dedicated presentation layer using one or more Web UI technologies. Depending on the UI technology, the generated content may have a very specific structure. It may also contain some additional information (meta-information) providing details about the data being displayed and the nature of the UI control used to show it to the end user.

This meta-information is typically used by CBTA to generate comprehensive test scripts. Based on this meta-information, CBTA can for instance, replace a generic action (such as a mouse-click) by a step performing a row selection (using the CBTA_WEB_SelectRow component).

Note

In general, the term UI control is used when targeting something that the end user can see – such as a button, a checkbox, etc.

The term UI element refers to something which is not necessarily visible or something the end user is not aware of. UI controls are most of the time an aggregation of several UI elements.
Web Dynpro Controls

The HTML content generated for a Web Dynpro Applications includes additional information coming from the underlying Unified Rendering Light Speed framework. In other words, each UI control includes a collection of Light Speed attributes that CBTA can use at runtime.

- Some of these attributes can be specified in the URI when searching for a Web Dynpro controls.
- The “ls.” prefix (for Light Speed) avoids conflicts between web control attributes and HTML attributes. For instance, `ls.id` is used instead of `id` when targeting Web Dynpro controls.
- CBTA discovers at runtime the attributes exposed by the targeted UI control (or UI element). This list of attributes is therefore dynamic and may differ depending on the UI control nature.

```
Note

For Web Dynpro Applications, the ls.id attribute can be used only when the so-called “stable ID” mode is enabled. This mode is activated by default when the session is started from TCE.
```

![Figure 32: Light Speed Attributes](image)

Light Speed Data

Do not confuse Light Speed Attributes and Light Speed Data. Only the first ones can be specified in the URI.

```
Note

Light Speed Data can be used when defining checkpoints. For instance, you may check whether a control can be modified by checking its ls.data.enabled attribute.
```
Web GUI Controls

SAP GUI for HTML maps SAP screens to HTML pages. Thus, the content of SAP GUI transactions can be shown in the browser as a regular HTML content, which is generated dynamically.

When starting the SAP GUI transaction in a Web browser, each SAP GUI control is converted to a Web GUI control using the **Unified Rendering Light Speed** framework.

The main difference, with Web Dynpro controls, is that the `ls.id` attribute generated for Web GUI controls is not stable. In other words, there is no stable ID mode for SAP GUI transactions. The alternative is to use another information exposed as the `ls.sid` attribute.

**Example**

Like shown below, the URI generated for Web GUI controls includes this information by default.

![Figure 33: URI for Web GUI Controls](image)

Java Web Dynpro Controls

**Java Web Dynpro Controls (from SAP NetWeaver 7.31)**

Since version 7.31 of SAP NetWeaver, Java Web Dynpro Applications are (like Web Dynpro ABAP) built on top of the Unified rendering Light Speed framework. They benefit from Light Speed attributes, and the URIs generated when testing them also use the "ls." prefix.

**Java Web Dynpro Controls (before SAP NetWeaver 7.31)**

Java Web Dynpro applications developed with former releases (before version 7.31) for SAP NetWeaver do not use the Unified rendering Light Speed framework. The "wd4j." prefix is used when targeting Java Web Dynpro controls.
WebCUIF Controls (for SAP CRM Web Applications)

CRM Web applications are based on the WebCUIF framework.

When the test mode is enabled, the generated HTML content includes additional information via a collection of attributes which is associated to the UI control. The “crm.” Prefix is the one used in the URI when targeting CRM UI elements.

See CRM Web UI Elements for more information.

SAP UI5 and FIORI Controls

Modern applications are now based on SAP UI5 and its FIORI extension. When testing these web applications, the generated URIs may use the “ui5.” prefix.

SAP UI5 Data

Unlike Web Dynpro applications, SAP UI5 Data can be used in URIs targeting SAP UI5 controls. This is done by default when the tool detects that the generated ID is not stable.

This is typically the case with the Fiori Tiles; they are identified using their type, their title and their subtitle (if any).

ui5.type=sap.m.GenericTile; ui5.data.title=Test Suite; ui5.data.subtitle=Overview

One may use the Object Spy to retrieve this information as shown below.
Figure 34: SAP UI5 Attributes
8.2 URI Resolution Strategies

UI elements can be found by type, attributes and properties. The default resolution strategy relies on the id or the name attribute (if any).

Complex scenarios can require searching for UI elements differently. The URI syntax provides several alternatives and allows searching by text (innerText), value, position (index), etc. Each alternative has constraints. It is for instance mandatory, when searching by index, to specify at least the tag of the HTML element.

Searching by ID

Searching by ID is preferred.

Web example targeting an ANCHOR element <A>:

```
tag=A; id=WDR_TEST_UI_ELEMENTS.ID_0001:MAIN.PERFORM_UPDATE
```

Web Dynpro ABAP example targeting a button (B):

```
ls.type=B; ls.tag=A; ls.id=WDR_TEST_UI_ELEMENTS.ID_0001:MAIN.PERFORM_UPDATE
```

Web Dynpro ABAP example targeting a button (B) in the WorkArea frame:

```
frameId=WorkArea; ls.type=B; ls.tag=A; ls.id=WDR_TEST_UI_ELEMENTS.ID_0001:MAIN.PERFORM_UPDATE
```

Searching by Name

Searching by name is not preferred. CBTA uses the name attribute when the UI Element has no ID. This is not very frequent.

Searching UI Elements by InnerText

Searching using the text visible to the end user is not recommended because it may lead to unpredictable results when, for instance, the texts are changed or translated. However, CBTA uses this approach when the targeted UI element has no ID and no NAME.

Example targeting a SPAN HTML element visible as “Apply Changes” to the end user:

```
tag=SPAN; innerText=Apply Changes
```

The innerText URI attribute can be combined with other attributes to avoid potential conflicts with other UI elements with the same text.

Example targeting a SPAN HTML element child of an ANCHOR:

```
tag=SPAN; innerText=Apply Changes; parentTag=A
```
Searching by Index

Searching by index is possible but not recommended, because it is sensitive to changes made by developers.

Example targeting a 3rd SPAN HTML element child of an ANCHOR:

```plaintext
tag=SPAN; innerText=Apply Changes; parentTag=A; index=3
```

URI Resolution Ambiguities

The ID and NAME should be unique within an HTML document, but when page composition is used, the same content can be embedded twice in two different frames, and this may lead to situations in which two UI elements have the same ID (in two different contexts). For test automation, these conflicts are very challenging, and a different strategy to search for the UI element is then mandatory.

The same applies if the UI element has no ID and no NAME.

Here is an example of a BSP sample in which three UI elements have no ID and the same text displayed:

![Example URI Resolution Ambiguities](image)

Figure 35: URI Resolution Ambiguities

To select the 3rd UI element (in red in the above screenshot), the URI must be adapted to resolve the conflicts.

Example searching for 3rd SPAN HTML Element:

```plaintext
tag=SPAN; innerText=text3; index=3
```

The Object Spy can indicate that only two of the three UI elements are links; their parent HTML element is an ANCHOR. The parentTag attribute can be used to eliminate the first one, and the UI element we want to select is now in the 2nd position.
As shown here, searching by index is not the best approach and may lead to unpredictable results when the UI of the application changes. It is better to search for a parent element with a stable ID and drill down to the child element using an URI fragment.

With Internet Explorer, the Microsoft Developer Tool (F12) lets you visualize the HTML content and go through the hierarchy of HTML elements.

In the example below, the ANCHOR HTML element id is `myBreadCrumb2`.

The following URI, with two fragments, can be used instead of using the index. This is more complex, but more stable, because the test script is less sensitive to minor changes that application developers may make.

**Searching using Regular Expressions**

The URI resolution supports regular expressions for some of the URI attributes. Regular expressions are activated by the operator: "~=

Example targeting an ANCHOR HTML element whose text starts with "Sales order 123456 has been created"
Make sure, when using regular expressions, that syntax is correct.

1. **Recommendation**

   *Regular expressions can be checked online; a lot of websites provide this feature.*

In this example, the regular expression specifies the following criteria:

- ^Sales order \(\rightarrow\) starts with "Sales order".
- \([0-9]\+\) \(\rightarrow\) matches at least one digit repeated n times.
- has been created$ \(\rightarrow\) ends with "has been created"

### Searching by Value

For complex scenarios which display dynamic content, searching by ID is not always the best option. Searching by value can, for example, be the best way of finding a value in a table.

Example of searching for the 3\(^{rd}\) INPUT HTML element whose value starts with:

- "ABAP Application Server: "

```plaintext
ls.type=I; tag=INPUT; value:\~=^ABAP Application Server: ; index=3
```

In this example, the URI combines web attributes (tag and value) with Light Speed attributes (ls.type). Combining attributes of different types is supported and recommended. The presence of the `ls.type` attribute indicates that the Unified Rendering Light Speed framework is the one used by the application being tested. Such information is important and used by the test player when waiting for the page availability (document readiness) before performing any action.

### Searching by Title

Some HTML elements may have a title attribute declaring the text displayed in the tooltip. Searching by title makes sense when the innerText or the value attributes are not set.

Example of searching for an INPUT HTML element whose value starts with "ABAP":

```plaintext
ls.type=I; tag=INPUT; title:\~=^ABAP
```

### Searching by using several HTML attributes

There are situations in which the HTML element cannot be identified uniquely by a single HTML attribute. The URI syntax lets you define the criteria using several HTML attributes, by using the `html` prefix.

It is therefore possible to search for an INPUT element matching the following criteria:

- the value HTML attribute starts with "ABAP"
• the auto-complete HTML attribute ends with "off".

Example:

```
ls.type=I; tag=INPUT; html.value~="ABAP; html.autocomplete~="off$; index=2
```

The "html." prefix can be used for all HTML attributes, including the ID and the name.

Example of checking for the ID using a regular expression:

```
ls.type=C; tag=INPUT; html.id~="WDR_TEST_UI_ELEMENTS\._ID_[0-9]+:MAIN\._CI_ENABLED_ATTR$
```

The "html." prefix can also be used with the innerText property. The following syntax is also supported:

```
ls.type=LN; tag=A; html.innerText~="Documentation; html.href~="CBTA/DOCUMENTATION$
```

*All work and no play makes Jack a dull boy*
8.3 Web UI Technology – Action Components

This section describes components that can perform a general action on any underlying UI technology.

**CBTA_WEB_A_CaptureScreen**

Technical Name: Actions\CaptureScreen

The "CaptureScreen" component captures a screenshot of the active browser window.

- The persisted screenshot will be in the execution report.

**Component Parameters**

**URI**

The URI is optional. It can be specified to capture a screenshot of a different window (when several windows are displayed).

**OPTIONS**

/e – (for element) – When a URI is specified, the component captures only the element that the URI identifies (instead of the whole window).

**Known Limitations**

The screenshot capture does not work when the computer is locked or a screen saver is being displayed.
CBTA_WEB_A_GetMessageParams

Technical Name: Actions\GetMessageParameters

This component retrieves information from application messages.

Most applications use a message area to inform the end user about the status of the latest operation. The message area may display messages of different types, like Errors, Warnings, Information, etc.

This component searches the message area for messages, using a pattern, and extracts information from the text being displayed.

Component Parameters

URI (optional)
The URI identifying the message area.

- If empty, the component searches for the message in the main document of the main Internet Explorer window.
- The URI syntax may vary depending on the underlying UI technology. Use the Object Spy to retrieve this information.

MESSAGEPATTERN
- The pattern that the message must match.

OPTION (optional)
- /u (uppercase) - to ignore the case when using the pattern

Output Parameters

MESSAGETEXT
- Output parameter providing the full message text.

MESSAGEPARAMETER1
- Output parameter providing the fragment which matches the placeholder {1}.

MESSAGEPARAMETER2
- Output parameter providing the fragment which matches the placeholder {2}.

MESSAGEPARAMETER3
- Output parameter providing the fragment which matches the placeholder {3}.

MESSAGEPARAMETER4
- Output parameter providing the fragment which matches the placeholder {4}.

Explanations

The syntax of the Message Pattern may include placeholders to extract some words from the text being displayed by the application.
The following placeholders exist:

- `{?}` ➔ Placeholder matching one or more words. Use this placeholder to ignore part of the text.
- `{1}` ➔ Placeholder matching one or more words and exposing them via the MessageParameter1.
- `{2}` ➔ Placeholder matching one or more words and exposing them via the MessageParameter2.
- `{3}` ➔ Placeholder matching one or more words and exposing them via the MessageParameter3.
- `{4}` ➔ Placeholder matching one or more words and exposing them via the MessageParameter4.

**Typical Example**

The screenshot below is an example application message.

![Shopping Cart Message](Figure 37: CBTA_WEB_A_GetMessageParams Example)

One may need to retrieve this information to pass it to the subsequent steps of the test script. The *Message Pattern* to retrieve this number would be:

- Shopping cart “[?]” with number `{1}` saved successfully

In this pattern, the words providing information about the user and the date are ignored and the *Shopping Cart Number* is exposed via the output parameter: `MessageParameter1`.

It would also have been possible to retrieve the information using the following syntax:

- Shopping cart “`{1} {2} {3}`” with number `{4}` saved successfully

In this second example:

- `MessageParameter1` ➔ TESTER_01
- `MessageParameter2` ➔ 20.09.2013
- `MessageParameter3` ➔ 22:02
- `MessageParameter4` ➔ 1000133807

**CBTA_WEB_A_ExecuteStatement**

Technical Name: Actions\ExecuteStatement

The “ExecuteStatement” component calls custom functions or subroutines created by the runtime library manager.

**Component Parameters**

**LIBRARY**

The Library parameter is the relative path to the library which contains the function to execute.
**STATEMENT**

The Statement parameter provides the instruction to be executed.

**OPTIONS**

Reserved for future use.

**Note**

- The statement specified will typically invoke a subroutine or a function, and will be executed by the VB script interpreter. Ensure the syntax of the statement is correct.
- Like all default components, the **STATEMENT** parameter can use tokens to retrieve values from the execution context.
CBTA_WEB_A_Invoke Function

Technical Name: Actions\InvokeFunction

The "InvokeFunction" component is similar to the “ExecuteStatement” component, and calls custom functions created by the runtime library manager.

The main difference is that the number of parameters passed to the custom function is fixed.

- The advantage is that the caller does not have to follow VB script syntax (the component builds the statement on its own)
- The disadvantage is that the invoked function must have 5 input parameters. Other functions cannot be called by this component.

Component Parameters

**LIBRARY**
- The Library parameter is the relative path, from the CBASE folder, of the library which contains the statement to execute.

**FUNCTIONNAME**
- This parameter specifies the name of the function to be executed.

**FUNCTION PARAMETERS**
- The function called using this component must have five parameters.

**PARAMETER1**
- Value of the first input parameter passed to the custom function

**PARAMETER2**
- Value of the second input parameter passed to the custom function

**PARAMETER3**
- Value of the third input parameter passed to the custom function

**PARAMETER4**
- Value of the fourth input parameter passed to the custom function

**OPTIONS**
- Value of the fifth input parameter passed to the custom function

Output Parameter

The component has an Output parameter that receives the value returned by the custom function, which can be used by subsequent steps.

Explanations

The component implementation resolves each token before passing the parameter values to the custom function. It is, for example, possible to pass the date using the %today% token in any of the parameters.
The custom function receives *Null* as parameter value when the parameter is empty. The `%blank%` token passes an empty string.

**CBTA_WEB_A_CloseWindow**

Technical Name: Actions\CloseWindow

This component closes the browser window.

**Component Parameters**

**URI**

- This URI identifies the window to be closed.

**CBTA_WEB_A_LogOff**

Technical Name: Actions\LogOff

This component logs the user out by clicking on "Log Off", and closes the browser window.

**Component Parameters**

**URI**

- This URI identifies the UI element (button or link) that logs the user out.
8.4 Web UI Technology – Generic Components

This section describes the components that target a UI element.

CBTA_WEB_CheckAttribute

Technical Name: Controls\CheckAttribute

This component checks attributes selected by the Check Picker when recording scenarios. It verifies whether the value of an HTML attribute. The component fails if the actual value does not match the expected value.

Component Parameters

URI

- The uniform resource identifier of the targeted HTML element.

ATTRIBUTE_NAME

- The name of the HTML attribute to be checked.

OPERATOR

- The Boolean operator which compares the actual value with the expected value. See the “Checkpoint Operators” section for more details.

EXPECTED_VALUE

- The expected value.

OPTIONS

- Some options can be set to perform some type conversions before doing the comparison. See the “Checkpoint Options” section for more details.
- Some other options can be used to define asynchronous checkpoints. Refer to section Options to define an asynchronous checkpoint for more details.

Component Output

OUTPUT

- This component has an output parameter which receives the value of the attribute, which subsequent components can use as input parameters.
**CBTA_WEB_CheckProperty**

Technical Name: Controls\CheckProperty

This component checks properties selected by the Check Picker when recording scenarios.

It verifies the value of an HTML property. The component fails if the actual value does not match the expected value.

**Component Parameters**

**URI**

- The uniform resource identifier of the targeted HTML element.

**PropertyName**

- The name of the HTML property to be checked.
- For UI technologies based on the Unified Rendering Light Speed layer (like Web Dynpro ABAP) the PropertyName can also refer to Light Speed Data, using the prefix “ls.data.”.

**Operator**

- The Boolean operator which compares the actual value with the expected value. See the “Checkpoint Operators” section for more details.

**ExpectedValue**

- The expected value. For boolean properties (such as “exist”), the expected value must be True or False.

**Options**

- Some option can be set to perform some type conversions before doing the comparison. See the “Checkpoint Options” section for more details.
- Some other options can be used to define asynchronous checkpoints. Refer to section Options to define an asynchronous checkpoint for more details.

**Explanations**

For HTML elements displaying a text, the innerText property provides the displayed information. For HTML elements such as input fields, the innerText property is not relevant. The value attribute provides the information, and the CBTA_WEB_CheckAttribute component should be used instead.

The property name can be set to “exist” to check whether the UI element (identified by the URI) exists in the HTML content.
**CBTA_WEB_Click**

Technical Name: Controls\Click

This component emulates a click on an HTML element.

**Component Parameters**

**URI**

- The uniform resource identifier of the targeted HTML element.

**Known Limitations**

This component cannot perform a double-click or a mouse click and keyboard combination (like Ctrl + Click). For such use case, a custom function might be necessary.

**CBTA_WEB_GetAttribute**

Technical Name: Controls\GetAttribute

This component retrieves the value exposed by UI elements via HTML attributes.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted HTML element.

**ATTRIBUTE_NAME**

- the name of the HTML attribute to retrieve the value from.

**EXPECTED_VALUE (OPTIONAL)**

- the expected value.

**TARGET_FIELD**

- This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to Getter Components section.

**Component Output**

**OUTPUT**

- This component has an output parameter which receives the value of the attribute. The subsequent components can use its value as input parameters.
**CBTA_Web_GetProperty**

Technical Name: Controls\GetProperty

This component retrieves the value exposed by UI elements via their HTML properties.

**Component Parameters**

- **URI**
  - the uniform resource identifier of the targeted HTML element.

- **PROPERTY NAME**
  - the name of the HTML property from which to retrieve the value.

- **EXPECTED VALUE**
  - the expected value.

- **TARGET FIELD**
  - This parameter can be used to persist the retrieved value in the CBTA Execution Context. For more details, refer to [Getter Components](#) section.

**Component Output**

- **OUTPUT**
  - This component has an output parameter which receives the value of the property. The subsequent components can use its value as input parameters.

**CBTA_WEB_OpenInputHelp**

Technical Name: Controls\OpenInputHelp

This component emulates the click on the “Value Help” button which opens a popup when pressing the F4 key.

**Component Parameters**

- **URI**
  - the uniform resource identifier of the targeted HTML input element.

**CBTA_WEB_OpenContextMenu**

Technical Name: Controls\OpenContextMenu

This component emulates the mouse click opening the contextual menu associated with a UI element.

**Component Parameters**

- **URI**
  - the uniform resource identifier of the targeted UI element.

**CBTA_WEB_PressKey**

Technical Name: Controls\PressKey
This component emulates a keystroke targeting an HTML element.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted HTML element.

**KEYCODE**
- the code identifying the key to press.

**Explanations**

The code is a string identification key; for instance: ENTER, F3, UP, DOWN. Keystroke combination is possible - e.g.: CTRL+ENTER, SHIFT+F3 or ALT+S.

Most common keyboard actions, such as those for navigation (e.g.: TAB) are not used in tests.

**CBTA_WEB_SelectMenuItem**

Technical Name: Controls\SelectMenuItem

This component emulates the selection of an item in the context menu associated with a UI Element. You can use the CBTA_WEB_OpenContextMenu component to open the context menu, and then perform a regular click on the child element.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted UI element.

**ITEM**
- the ID of the menu item

**OPTIONS**
- Reserved for future use

**CBTA_WEB_SetAttribute**

Technical Name: Controls\SetAttribute

This component sets the value of an HTML attribute of a UI element.

**Component Parameters**

**URI**
- the uniform resource identifier of the targeted HTML element.

**ATTRIBUTENAME**
- the name of the HTML attribute to modify.

**THEVALUE**
- the new value of the attribute.
**CBTA_WEB_SetFocus**

Technical Name: Controls\SetFocus

This component puts the focus on an HTML element.

*Component Parameters*

**URI**

- the uniform resource identifier of the targeted HTML element.

**CBTA_WEB_SetProperty**

Technical Name: Controls\SetProperty

This component sets the value of an HTML property of a UI element.

*Component Parameters*

**URI**

- the uniform resource identifier of the targeted HTML element.

**PROPERTYNAME**

- the name of the HTML property to modify.

**THEVALUE**

- the new value of the property.

**CBTA_WEB_SetState**

Technical Name: Controls\SetValue

This component sets the state of a UI element.

*Component Parameters*

**URI**

- the uniform resource identifier of the targeted UI element.

**THEVALUE**

- the new state of the UI element.

**CBTA_WEB_SetValue**

Technical Name: Controls\SetValue

This component sets the value of a UI element.

*Component Parameters*

**URI**

- the uniform resource identifier of the targeted UI element.
**THE VALUE**

- the new value of the UI element.

**CBTA_WEB_SelectRow**

Technical Name: Controls\SelectRow

This component emulates a click on a cell to select a line in tables.

**Component Parameters**

**URI**

- the uniform resource identifier of the targeted cell identifying the row in the table.

**CBTA_WEB_SelectTab**

Technical Name: Controls\SelectTab

This component emulates the selection of a tab.

**Component Parameters**

**URI**

- the uniform resource identifier of the tab to select.
8.5 Web UI – Test Automation Challenges

Handling of Internet Explorer Windows

Scenarios where several Internet Explorer windows are opened are properly recorded and the test player identifies them at runtime using the `windowId` URI attribute.

- The `windowId` for the main window is implicit (i.e.: `windowId=0` for the main window)
- The `windowId` gets incremented each time a new window is opened (i.e.: `windowId=1` for the first child window of the main window)
- Nested window hierarchy is supported (i.e. `windowId=1.2` for the second child window of the first child of the main window)

**Alternative to using the window ID**

For some scenarios it might be easier to search for a particular window using its title (or part of it). This is now possible using the `windowTitle` URI attribute.

- `windowTitle` can be used instead of the `windowId`
- Regular expressions are supported

**Examples**

The two URI(s) below search for the same UI element in a child window. The difference is that the second one uses a regular expression to search for the internet explorer window using its title.

```
label=Add Item; windowId=1; ls.type=B; ls.tag=A; ls.id=ADD_ITEM
label=Add Item; windowTitle=~^Shopping Cart Creation; ls.type=B; ls.tag=A; ls.id=ADD_ITEM
```

Note that `~=` operator is used to inform the URI resolver that the value specified here is a regular expression. This example searches for a title starting with "Shopping Cart Creation"
Internet Explorer Security Popups

Security popups are native windows asking for confirmation before performing an action. Most of the actions made against security popups are not recorded and cannot be automated. The best practice is to prevent them by a proper configuration of both the browser and the application being tested.

Notification Bar

Unfortunately, this is not always possible. This is typically the case for the notification bar asking for confirmation when operation like a file download occurs:

![Internet Explorer Security Popups](image)

Figure 38: Internet Explorer Security Popups

- Actions made against the notification bar are not recorded
- The generated test script must be adapted manually

Specific URI Attribute – ie.control

A new URI attribute has been introduced to provide the ability to perform actions against the notification bar.

- Uri attribute ➔ ie.control

Uri Attribute Values

The supported values are the ones below:

- ie.control=NotificationBar.Open
- ie.control=NotificationBar.Save
- ie.control=NotificationBar.SaveAs
- ie.control=NotificationBar.Save&Open
- ie.control=NotificationBar.Cancel
How to use it

Component CBTA_WEB_CLICK can be used to perform a mouse click like shown in the screenshots below where the application lets the user download a file from the server.

![Security Popup - CBTA_WEB_Click Example](image1)

**Input Parameter**

The URI parameter uses here the new URI attribute to save the downloaded file.

![Security Popup - URI of the Save Button](image2)
9 Web Dynpro / Light Speed

Unified Rendering Light Speed is a common layer for most of the SAP applications developed using:

- Web Dynpro ABAP,
- Web Dynpro Java,
- Web GUI

The default components described in this section have been introduced to benefit from this Light Speed layer and make it easier to test those applications.

9.1 Light Speed - Action Components

*CBTA_LS_A_GetMessageParams*

Not yet available.

Use the generic component *CBTA_WEB_A_GetMessageParams* instead.
9.2 Light Speed - Control Components

**CBTA_LS_T_FindRow**

Technical Name: Controls\Table\FindRowByContent

This component searches for a row by content.

**Component Parameters**

**URI**

Specifies the uniform resource identifier of the parent table. For more information, refer to the URI syntax on page 16.

**COLUMN TITLE**

Specifies the title of the column. The title is the information visible to the end user.

When there is no column title the technical name of the column can be used instead and together with the /ColumnName option.

**OPERATOR**

This operator is a Boolean operator to compare the actual value with the expected value. Refer to the section “Checkpoint Operators” for more details.

**CELL CONTENT**

Specifies the value to search for.

**OPTIONS**

There are various ways to search for the row.

**Search Options**

<table>
<thead>
<tr>
<th>/ColumnName</th>
<th>This option must be used when there is no column title. With this option the technical name of the column can be used instead of the column title. The value of the ColumnTitle parameter must be determined using the Object Spy.</th>
</tr>
</thead>
</table>

**Options to Triggering Actions**

The options below define the action to perform once the row has been found.

<table>
<thead>
<tr>
<th>/Select</th>
<th>When this option is specified the first row matching the criteria is selected. This to avoid having to select the row using the CBTA_WEB_SELECTROW component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Expand</td>
<td>This option let you expand the selected row. This only makes sense when the row is a tree node.</td>
</tr>
</tbody>
</table>
**Scrolling Options**

By default, the component implementation does not scroll. It only checks the content of the rows being displayed in the current HTML document.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Scroll</td>
<td>This option enables an implicit scroll mechanism. If the row is not found on the current page the component implementation will try to scroll down and search for the row in the subsequent pages.</td>
</tr>
<tr>
<td>/ScrollTop</td>
<td>By default, the component implementation start searching use the current page. The /ScrollTop option can be used to start searching from the top of the table – In other words it will first scroll up to make sure the first row is visible.</td>
</tr>
</tbody>
</table>

**Type Conversion Options**

Some other options can be used to alter or convert both the actual cell value and the expected cell value before comparing them. Note that options for converting values are lowercase.

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

**Component Output**

**OUTPUT**

This component has an output parameter which receives the row number. The subsequent steps may rely on either the output parameter or the %Output% token to reuse the information.

(new 3.0.8) The %Row% token provides the same information.

**Example**

The screenshot below shows a typical situation where the table is used to display a hierarchy of nodes and no column title is being displayed. The test might require scrolling down through the table content to search for the `Z_DEMO123` node and expand it.
Explanation

The **CBTA_LS_T_FindRow** component allows performing all operations in one shot using the following input parameters:

- **Uri** ⇒ ls.type=ST; ls.tag=TABLE;
  ls.id=AGS_TST_SUT_MANAGEMENT.ID_0001:V_SUT_MANAGEMENT.TABLE_SDC_RELATION
- **ColumnTitle** ⇒ SDC_ENTRY_TEXT
- **Operator** ⇒ =
- **CellContent** ⇒ Z_DEMO123
- **Options** ⇒ /ColumnName /ScrollTop /Scroll /Select /Expand
In this example there is no column title. The technical column name is used instead. For Web Dynpro applications the technical column name can be retrieved using the Object Spy feature.

As shown below, the id of the cell includes this information.
Note that the Object Spy can also be used to retrieve the URI of the parent table. You may spy one of the cells and open the hierarchy of attributes to find the URI like shown below:

Figure 44: Object Spy - URI of the Parent Table
CBTA_LS_T_SetFilterValue

Technical Name: Controls\Table\SetFilterValue

This component sets the value of a cell in the filter row of a table.

Component Parameters

URI
- the uniform resource identifier of the table.

COLUMNTITLE
- the title of the column.

CELLCONTENT
- the new value of the cell.

ROW
- the row number.

OPTIONS
- Reserved for future use.

Notes

The uniform resource identifier of the table can be retrieved by copying the ‘Parent Table Uri’ property of the ‘Light Speed Context’ node of the cell, using the Object Spy.

This component activates the filter row in the specified table, if needed, and validates the filter after having set the filter value.
CBTA_LS_T_SetFilterValues

Technical Name: Controls\Table\SetFilterValues

This component sets the value of cells in the filter row of a table.

Component Parameters

URI
- the uniform resource identifier of the table.

COLUMNVALUEPAIRS
- the title of a column and new value of the cell.

OPTIONS
- Reserved for future use.

ColumnValuePairs Syntax

The column title and the cell value are separated by an equals ‘=’. Each column title and cell value pair is separated by a semicolon and a space ‘; ’.

For example:

```text
  c1=value1; c2=value2; c3=value3
```

Notes

The uniform resource identifier of the table can be retrieved by copying the 'Parent Table Uri' property of the 'Light Speed Context' node of the cell, using the Object Spy.

This component activates the filter row in the specified table, if needed, and validates the filter after having set the filter values.
**CBTA_LS_T_SetCellValue**

Technical Name: Controls\Table\SetCellValue

This component sets the value of cell in a table.

**Component Parameters**

**Uri**
- the uniform resource identifier of the table.

**COLUMNTITLE**
- the title of the column.

**CELLCONTENT**
- the new value of the cell.

**ROW**
- the row number.

**OPTIONS**
- Reserved for future use.

**Note**

The uniform resource identifier of the table can be retrieved using the Object Spy. The information is exposed via the “Parent Table Uri” property of the “Light Speed Context”.
**CBTA_LS_T_SetCellValues**

Technical Name: Controls\Table\SetCellValues

This component is used to set value of cells in a row in a table.

**Component Parameters**

**URI**
- the uniform resource identifier of the table.

**COLUMNVALUEPAIRS**
- the list of title of column and new value of the cell.

**ROW**
- the row number.

**OPTIONS**
- Reserved for future use.

**Column Value Pairs Syntax**

The column title and the cell value are separated by an equals ‘=’. Each column title and cell value pair is separated by a semicolon and a space ‘; ’. For example:

| columntitle1=value1; columntitle2=value2; columntitle3=value3 |

**Note**

The uniform resource identifier of the table can be retrieved using the Object Spy. The information is exposed via the "Parent Table Uri" property of the "Light Speed Context".

---

**Web Dynpro / Light Speed**
10 SAP UI5 / FIORI

SAP UI5 is a UI framework used by modern applications such as the SAP Fiori apps.

Most of the time the test automation of SAP UI5 scenarios does not require specific components. In other words, the default components for Web UI technologies can be reused. Only a few default components are specific for the SAP UI5 technology. These components have a name starting with “CBTA_UI5_”.

10.1 SAP UI5 - Action Components

CBTA_UI5_A_GetMessage (new 3.0.8)

Technical Name: Actions\GetMessage

SAP UI5 applications may use a message container of type sap.m.MessagePopover to provide human readable feedback to the end user.

The CBTA_UI5_A_GetMessage component can be used to extract information from messages displayed in that particular message container. Messages are identified using their key. When no message key is defined (by the developer of the application), the component CBTA_UI5_A_GetMessageParams must be used instead.

Component Parameters

URI
- URI identifying the message container

MESSAGEKEY
- The key identifying the message.

EXPECTEDMESSAGETYPE
- The expected message type

Figure 45: SAP UI5 Message Container
How to use it?

The easier way to make use of this component is to define a **checkpoint** while recording the scenario. When a message is selected the component is automatically inserted into the recorded test script as shown in the screenshot below.

![Test Creation Wizard - Recording](image)

The recorder makes sure to select the component matching the context. It will use the **CBTA_UI5_A_GetMessage** component only if the **MessageKey** is set.

**Component Output**

The component searches the message container for the first message matching the specified **MessageKey**. It reports an error if the message is not found.

If the **ExpectedMessageType** parameter is specified, the component also checks whether the actual message type is the expected one.

If the message is found, the message parameters are collected, stored in the execution context, and returned as output parameters. The following output parameters are populated:

- **MessageType** – The actual message type
- **MessageText** – The complete text being displayed
- **MessageParameter1** – The parameter #1 of this message
- **MessageParameter1** – The parameter #2 of this message
- **MessageParameter1** – The parameter #3 of this message
- **MessageParameter1** – The parameter #4 of this message
**CBTA_UI5_A_GetMessageParams (new 3.0.8)**

Technical Name: Actions\GetMessageParameters

This component is equivalent to component **CBTA_WEB_A_GetMessageParams**.

It gets information from message containers that are used by the applications to provide human readable feedback to the end user.
10.2 SAP UI5 - Control Components

CBTA_UI5_T_FindRow (new 3.0.8)

Technical Name: Controls\Table\FindRowByContent

This component searches for a row by content.

Component Parameters

URI
Specifies the uniform resource identifier of the parent table. For more information, refer to the URI syntax on page 16.

COLUMNTITLE
ColumnTitle specifies the title of the column. The title is the information visible to the end user. When there is no column title the technical information associated to the column can be used instead. You must in that case use the /Binding option.

OPERATOR
This operator is a Boolean operator to compare the actual value with the expected value. Refer to the section “Checkpoint Operators” for more details.

CELLCONTENT
Specifies the value to search for.

OPTIONS
There are various ways to search for the row.

Search Options

<table>
<thead>
<tr>
<th>/Binding</th>
<th>This option must be used when there is no column title. With this option the technical information associated to the column can be used instead of the column title.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The value of this parameter must be determined using the Object Spy.</td>
</tr>
</tbody>
</table>

Options to Triggering Actions

The options below define the action to perform once the row has been found.

<table>
<thead>
<tr>
<th>/Select</th>
<th>When this option is specified the first row matching the criteria is selected. This to avoid having to select the row using the CBTA_WEB_SELECTROW component.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This option only makes sense for tables where the rows can be selected:</td>
</tr>
<tr>
<td></td>
<td>• sap.m.Table</td>
</tr>
<tr>
<td></td>
<td>• sap.ui.table.Table</td>
</tr>
</tbody>
</table>
### Scrolling Options

There are no scrolling options available for SAP UI5 applications.

### Type Conversion Options

Some other options can be used to alter or convert both the actual cell value and the expected cell value before comparing them. Note that options for converting values are lowercase.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/u</td>
<td>Both values are converted to upper-case before being compared</td>
</tr>
<tr>
<td>/t</td>
<td>Both values are trimmed before being compared</td>
</tr>
<tr>
<td>/i (int)</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 (or 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>

### Component Output

**OUTPUT**

This component has an output parameter which receives the row number. The subsequent steps may rely on either the output parameter or the %Output% token to reuse the information.

Two additional tokens can be used as output. (new 3.0.8)

- %Row% ➔ provides the index of the selected row
- %RowSuffix% ➔ provides the row suffix – this suffix is the one used by the IDs of the cells that the row contains.

Note that the row suffix is not always set. It makes sense only for a certain types of tables.
11 Runtime Library API

Customizing the runtime library is only useful if the tester can reuse the existing features of the CBASE, but SAP can change the content of the CBASE at any time, and introduce incompatibilities with the custom code.

To avoid backward compatibility issues, the custom code should only depend on public APIs. It should only invoke functions and subroutines that SAP guarantees to support in future releases.

This is the purpose of the CBTA class.

11.1 CBTA Class

The CBTA class is a helper class which makes useful functions available to custom code. The runtime library initializes the CBTA object.

Example of retrieving the SAP GUI Session object

```vba
Dim mySession
Set mySession = CBTA.GetSapGuiSession()
```

Example of capturing a screenshot of the SAP GUI active window

```vba
CBTA.CaptureScreen()
```

Example of writing feedback to the Execution Report

```vba
CBTA.Report CBTA.INFO, "Custom Code", "A feedback to the tester", ""
```

Public API of the CBTA Class

This CBTA API is a wrapper on top of the SAP GUI Scripting API. Most of the functions return references to SAP GUI Scripting objects. For the complete list of public properties and methods, refer to the official SAP GUI Scripting help (delivered with the SAP front end).

Public subroutines and functions of the CBTA Class are described below.

**Function GetSAPGUIConnection()**

The "GetSAPGUIConnection" function retrieves the SAP GUI connection.

**Return value**

The SAP GUI connection (SAP GUI Scripting object).
**Function GetSAPGUISession()**

The “GetSAPGUISession” function retrieves the SAP GUI session.

**Return value**

The SAP GUI session (SAP GUI Scripting object).

**Function GetControl( URI )**

The “GetControl” function retrieves a reference to an SAP GUI control, with which you can perform additional operations, depending on the nature of the control. Refer to the official SAP GUI Scripting help (delivered with the SAP Front End) for the complete list of public properties and operations.

**Parameters**

**URI**

The uniform resource identifier of the targeted object. For more information, refer to the [URI syntax on page 16](#).

**Return value**

A control, as documented in the SAP GUI Scripting help.

**Function ResolveParameterValue( Value )**

The “ResolveParameterValue” function resolves the parameter value by interpreting the tokens.

**Parameters**

**PARAMETER**

The “Parameter” parameter is a string that might include tokens (c.f.: Execution Context).

**Return value**

The resolved value (string).

**Note**

If the “Parameter” parameter has not been, or cannot be, resolved, it returns unchanged (string).

**Sub Report( Severity, Topic, Message, Options )**

The “Report” procedure adds a step to the execution report (ReportLog.xml).

**Parameters**

**SEVERITY**

Possible values are:

- CBTA.INFO
- CBTA.DONE
- CBTA.PASSED
- CBTA.WARNING
The "Topic" parameter is the step summary (string)

**MESSAGE**
The "Message" parameter is the step description (string)

**OPTIONS**
Reserved for future use

**Sub Log( message )**
The "Log" procedure logs in the debug file (DebugLog.txt)

**Parameters**

**MESSAGE**
The "Message" parameter is a log entry (string)

**Sub CaptureScreen()**
The subroutine is equivalent to the "CaptureScreen" component. It takes a screenshot of the window of the current SAP GUI session.

- The persisted screenshot will be in the execution report.
- Default location for screenshot is "%TEMP%\SAP\CBTA\Logs\<TestName>\Images"

**Known Limitations**
The screenshot is captured by the Hardcopy method of the SAP GUI Scripting API, which does not work properly if the computer is locked or the screen saver is running.

**Sub Wait( milliseconds )**
The "Wait" procedure stops the execution for specified length of time.

**Parameters**

**MILLISECONDS**
The "Milliseconds" parameter is the waiting time. The time is in milliseconds (number).

**Sub LoadLibrary( Library )**
The "LoadLibrary" procedure loads the specified library.

**Parameters**

**LIBRARY**
The "Library" parameter is the relative path, from the CBASE folder path, of the library to load.
12 References

12.1 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 – Object Spy</td>
</tr>
<tr>
<td>CBTA – Test Recorder</td>
</tr>
<tr>
<td>CBTA – Runtime Library Manager</td>
</tr>
<tr>
<td>CBTA – Custom Code Patterns</td>
</tr>
<tr>
<td>CBTA – Query API</td>
</tr>
</tbody>
</table>

12.2 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>
13 Table of Figures

Figure 1: Test Repository Tile......................................................................................................................... 13
Figure 2: Launch CBTA from TCE .................................................................................................................... 14
Figure 3: Starting the Object Spy from TCE .................................................................................................... 15
Figure 4: Page Composition Example ............................................................................................................ 17
Figure 5: Checkpoint Defined while Recording ............................................................................................ 22
Figure 6: Checkpoint Operators ...................................................................................................................... 24
Figure 7: Unit Property of a SAP UI5 tile ........................................................................................................ 25
Figure 8: Check using a Regular Expression .................................................................................................. 26
Figure 9: Input Parameters for an Asynchronous Checkpoint ....................................................................... 28
Figure 10: Tokens shown in the Execution Report ......................................................................................... 29
Figure 11: CBTA Keywords - IF / ELSE / ENDIF .......................................................................................... 36
Figure 12: DO / LOOP Iteration Example .................................................................................................... 39
Figure 13: FOR / NEXT Iteration Example .................................................................................................... 41
Figure 14: CBTA_GUI_T_SelectColMenuitem Example .............................................................................. 106
Figure 15: CBTA_GUI_T_SelectItem Example ............................................................................................... 109
Figure 16: CBTA_GUI_T_UnselectColumn Example ....................................................................................... 112
Figure 17: CBTA_GUI_T_SelectColMenuitem Example .............................................................................. 113
Figure 18: Object Spy - GuiTree Information ................................................................................................. 114
Figure 19: SAP GUI Scripting Recorder ...................................................................................................... 115
Figure 20: SAP GUI Scripting Example ........................................................................................................ 116
Figure 21: Internet Explorer - File Upload Dialog .......................................................................................... 120
Figure 22: Runtime Library Manager - Code Assistant ................................................................................ 120
Figure 23: Runtime Library - Code Pattern Selection .................................................................................. 121
Figure 24: Calling the DoFileUpload custom function ................................................................................ 121
Figure 25: DoFileUpload Input Parameters .................................................................................................. 121
Figure 26: Example of an Embedded HTML Content .................................................................................. 123
Figure 27: SAP GUI Scripting API ............................................................................................................... 124
Figure 28: CBTA_CRM_A_LogOff Example ................................................................................................. 131
Figure 29: CBTA_CRM_A_ClosePopup Example ............................................................................................ 132
Figure 30: SAP CRM - Collapsible Buttons ................................................................................................. 139
Figure 31: SAP CRM - Transaction Type Selection ...................................................................................... 148
Figure 32: Light Speed Attributes ................................................................................................................ 153
Figure 33: URI for Web GUI Controls .......................................................................................................... 154
Figure 34: SAP UI5 Attributes ...................................................................................................................... 156
Figure 35: URI Resolution Ambiguities ...................................................................................................... 158
Figure 36: URI Ambiguities Example .......................................................................................................... 159
Figure 37: CBTA_WEB_A_GetMessageParams Example .............................................................................. 164
Figure 38: Internet Explorer Security Popups ............................................................................................... 176
Figure 39: Security Popup – CBTA_WEB_Click Example .......................................................................... 177
Figure 40: Security Popup - URI of the Save Button ..................................................................................... 177
Figure 41: CBTA_LS_T_FindRow Example .................................................................................................... 181
Figure 42: CBTA_LS_T_Window Input Parameters ........................................................................................ 182
Figure 43: Object Spy - Technical Column Name ........................................................................................ 182
Figure 44: Object Spy - URI of the Parent Table .......................................................................................... 183
Figure 45: SAP UI5 Message Container ...................................................................................................... 188
Figure 46: Getting Message Parameters while Recording ......................................................................... 189