## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td>2018-12-03</td>
<td>CBTA 3.0 SP11 Update</td>
</tr>
<tr>
<td>1.5</td>
<td>2018-05-15</td>
<td>CBTA 3.0 SP10 Update</td>
</tr>
<tr>
<td>1.4</td>
<td>2017-09-30</td>
<td>CBTA 3.0 SP9 Update</td>
</tr>
<tr>
<td>1.3</td>
<td>2017-03-01</td>
<td>CBTA 3.0 SP8 Update</td>
</tr>
<tr>
<td>1.2</td>
<td>2014-05-27</td>
<td>CBTA 3.0 SP2 Update</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Running a CBTA Test Script in Debug Mode** .......................................................... 4

2. **How-to start** ............................................................................................................ 5
   2.1 Add or remove breakpoint ............................................................................. 8
   2.2 Step Over .......................................................................................................... 9
   2.3 Run ..................................................................................................................... 9
   2.4 Stop Debugger .................................................................................................. 9
   2.5 Error Behavior ................................................................................................. 10
   2.6 Dynamic Report .............................................................................................. 11
   2.7 Launch Object Spy ......................................................................................... 12

3. **How to Connect an External Debugger to CBTA debugger** ............................ 13
   3.1 Preparation ...................................................................................................... 13
   3.2 Starting the external debugger ..................................................................... 15
1 Running a CBTA Test Script in Debug Mode

If a CBTA test script does not work, you can execute the components (test steps) one by one. Component-Based Test Automation offers a graphical overview of the test steps and displays the corresponding UI component attributes.

The CBTA debugger allows you to debug a test while remaining at the component level, without seeing the Visual Basic Scripting (VBS) code lines. Therefore, you see the actual state of the tested application UI, as well as input and output parameter values.

To examine the VBS code, you can use the Microsoft Visual Studio debugger. See Section 3 - How to Connect an External Debugger to CBTA debugger.
2 How-to start

In Solution Manager 7.2, use transaction SM_WORKCENTER to start the Solution Manager Fiori LaunchPad.

1. Select Test Suite group.
2. Open Test Repository (test configuration or test scripts)
3. In the Test Repository, select a CBTA test script, and choose Execute.

You go to the Start Options screen.
4. On the UI Control tab page, in the Mode for External Tool field, select Debug Mode.

5. Choose Execute.

The CBTA debugger is started. A graphical overview of the test steps is displayed.
The point hit of test script step is highlight the component box in yellow and blue arrow marker is add on component box.

2.1 Add or remove breakpoint

Two possibilities to add or remove a test step breakpoint:
- Select one component and choose the Toggle Breakpoint button
- Double click on one component

When a breakpoint is configured a red circle marker is add into component box selected.

- To change parameter values dynamically, double-click them in the table, edit the new value and press enter.
- To examine the code, choose the Start External Debugger pushbutton. See chapter Error! Reference source not found.

Shortcut key: F4
2.2 Step Over

In case you would move to the next component in your script, click on toolbar button Step Over.

CBTA debugger will execute the current action of default component before to stop test execution on next component.

**Shortcut key:** F6, F10

2.3 Run

In case you would move to the next breakpoint active in your script, click on toolbar button Run.

CBTA debugger will execute the all components until to stop test execution on next component. Has breakpoint marker.

**Shortcut key:** F5, F8

2.4 Stop Debugger

If you would stop the debugging session, click on toolbar button Stop. The test execution is stopped and the test report will be displayed. Into test report a message is available the test script has been stopped by debugger.
2.5 Error Behavior

In case of error during execution, the error message is displayed in the status message at the bottom of the CBTA debugger window.

More information can be seen by using the tooltip onto this status bar message (just leave your mouse some seconds without moving onto the error message to get the tooltip displayed).

Once an error happens, you need to press the Run button to end the execution and go back to the SAP GUI session (with the Start Options screen) to display the execution report.
2.6 Dynamic Report

The report tab dynamically shows the HTML type report being constructed after each component execution.
2.7 Launch Object Spy

To analyze some control display into the application in real time, you can launch the Object Spy application in clicking on toolbar button Object Spy.

The spied session is not opened by the Object Spy. You need to select into Object Spy session selector.
3 How to Connect an External Debugger to CBTA debugger

This chapter describes a step-by-step procedure to initialize and use Visual Studio as external debugger for CBTA test script. Visual Studio 2010 is the only supported external debugger.

3.1 Preparation

In Visual Studio, open the Tools/Attach to process menu.
Set the code type to Script:

Press *Select*, check the *Script* entry, then *Ok* on the little popup window, and finally press *Cancel* on the larger window.

This setting remains set even if you close Visual Studio.
3.2 Starting the external debugger

In the CBTA debugger window, choose the *External Debugger* button.

Visual Studio opens and shows a default Solution:
It is important to understand that there are now two debuggers on the same VBS code:

- First one is the CBTA debugger that is a COM object called by the VBS script. So it is not a real debugger but rather a dot net UI showing the current state of execution and also allowing communication with the program to get and set parameter and execution context values.
- The second one is the Visual Studio debugger (called external debugger). It is also attached to the same VBS script.

**Press the pause button in Visual Studio to stop on the next VBS instruction.**

The “eval” code in the Solution contains the CBTA CBASE function (library of functions, subs and variables).
The other file with the vbs extension holds the test script code. We can see the VBS interpreter program is wscript.exe.

**Choose Run or Step Over button in the CBTA debugger to give back execution to the VBS script. It is recommended to choose Run.**

Visual Studio pauses the execution to the next VBS instruction. This is located in the function used to communicate with the CBTA debugger.

It is now possible to use all the Visual Studio functions like Step Into, Step Over, Step Out, variable edition, call stack etc.
Please note that before each component execution, the CBTA debugger will be called again by the VBS script. If you have pressed the Step Over button or defined a breakpoint in the CBTA debugger, then it will block execution. If you chose Run, then it will not stop the execution.

The Visual Studio commands act as a lower level (VBS instruction) than the CBTA debugger (component level). Visual Studio leads the VBS script execution.

The CBTA debugger is called by the VBS script during execution.