**SAP Soultion Manager**

**Current Version :** SAP Logon 740

**Test Suite Use**: You can determine the scope of testing required for cross-system business processes, manage the tests centrally and execute the tests.

The test management has the following phases:

1. Determining the testing scope

When introducing new SAP solutions, the testing scope can be derived from the [**solution documentation**](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3ef70f5452e7024be10000000a44176d.html)**.** When changes are made to the system, for example activation of support packages), you can determine the critical business processes using a [**change impact analysis**](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/e7ae57e0291149adaca93a9dcce33abb.html)**.**

1. Test planning

You set up a test system, generate test data, develop manual or automatic test cases, create test plans, and assign test packages to the right testers.

1. Testing

You initiate the execution of manual and automatic tests and check the progress and results of the test.

1. Transferring the changes to production operation

# Solution Documentation

## Use

## SAP Solution Manager provides various tools and methods for implementing and maintaining both SAP and customer solutions. The **Solution Documentation** model is based on a hierarchical structure, which uses a solution as the single, central point of access for all Solution Documentation content.

Recommendation

As a starting point, familiarize yourself with the new terms and concepts related to process management in SAP Solution Manager. Even familiar terms already used in former releases may have different meaning in this release. See [Basic Terms and Concepts](https://help.sap.com/viewer/60943adf3ff44893b62c568bb8a87d17/7.2.07/en-US/0a65ed54e1e47254e10000000a441470.html).

## Prerequisites

**Configuration of Process Management in SAP Solution Manager Configuration**

In **SAP Solution Manager Configuration**, you have to do the following:

* You have to configure all applications-relevant sub scenarios in **Cross Scenario Configuration**.
* You have to configure all sub scenarios in **Mandatory Configuration**.

For a correct graphical representation of solutions, you have to configure the scenario **Infrastructure Preparation**, step **Enable Gateway Services**. In the list of gateway services for **Process Management** you have to activate the services AGS\_GBC\_ODATA\_BPMN\_SRV (Graphical Component BPMN Gateway Service), AGS\_GBC\_ODATA\_GOM\_SRV (GBC GOM Gateway Service), and PROCESSMANAGEMENT (SAP Solution Manager APIs for third party tool integration).

* You have to configure the scenario **Embedded Search** in order to establish the connection between Embedded Search and **TREX** or **SAP HANA**.
* You have to configure the process management in the scenario **Process Management**. Refer to the scenario documentation to learn more.

## Features

**Functional Highlights of the Solution Documentation**

The Solution Documentation provides you, among others, with functional highlights in the following areas:

* **Usability**

You use a modern browser-based user interface making the administration and documentation of SAP solutions simple, intuitive and business-focused, and you benefit from graphical process modelling.

* **Context-driven user interface**

There are almost no buttons in the user interface to add new elements. Just use the right mouse button at the location where you want to add the new element. The context menu offers all the functions that are allowed at the selected location or element.

* **Implicit Save of user entries**

You do not need to save your entries with a “Save” button. Every change of an element is saved automatically when you select another element. Optionally, or if you want to finish your work without selecting another element, you can save your work manually with the “Save” button.

* **Re-Use**

You can organize reusable documentation for business process steps and other entities such as transactions, development objects, and configuration activities in libraries. Reduce your content management effort by working on reusable documentation elements only once for multiple use in different end to end business processes.

* **Integration**

You can take advantage of the concept of Solution Documentation as a single source of truth for all content related to business processes. A solitary solution will support all application lifecycle phases from implementation through to operations. Do away with creating solutions altogether when they are not needed, for example for sole use of basic SAP services, such as EarlyWatch Alert.

* **Versioning**

You can simplify maintenance, new development and upgrades by using the version concept – branches. With this concept, you are part of the quantum leap in SAP release management.

* **Big Solutions**

You can handle solutions of all shapes and sizes with multi-level hierarchies without limitations on the size and hierarchical complexity.

* **Custom Extensions**

You can add extra structure levels and your own attributes by taking advantage of the flexible, model-based architecture of Solution Manager 7.2 to systematically extend SAP standards.

**How Solution Documentation works**

The Solution Documentation model is based on a hierarchical structure, which uses a unified **solution** concept as the single, central point of access for all Solution Documentation content. The solution comprises the description of your system landscape as well as the documentation of your business processes, interfaces, technical objects and your custom developments.

Thus, the **solution** concept stands in the centre of the Solution Documentation. It is the sum of a company’s system landscape, applications and processes. It acts as a container for versions of Solution Documentation, one of which is the production version. The terms **systems landscape**, **process** and **version** have the following meaning:

* **System Landscape**

Logical Component Groups are groups of Logical Components. Logical components point to actual technical systems.

* **Process**

A process is a set of logically related activities and executes functions in systems to achieve a business goal.

* **Version**

Each Solution Documentation version is a branch containing processes, libraries, and systems.

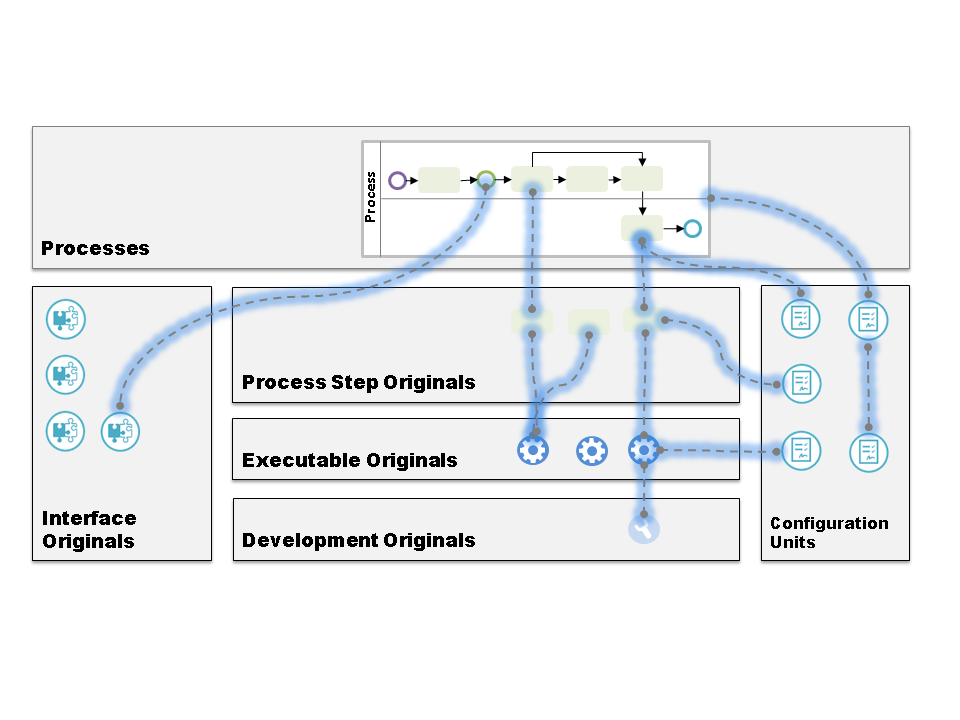
You organize your Solution Documentation into the following different areas:

* **Libraries** of reusable documentation for technical objects, such as development objects, executables, and configuration activities
* **Library of business process steps** where you can reuse the documentation from the other libraries
* **Library of interfaces** where you document the interfaces used by your end to end processes
* **End to End Business Processes** where you model your scenarios and processes by reusing the process steps, interfaces, and technical objects from the libraries,

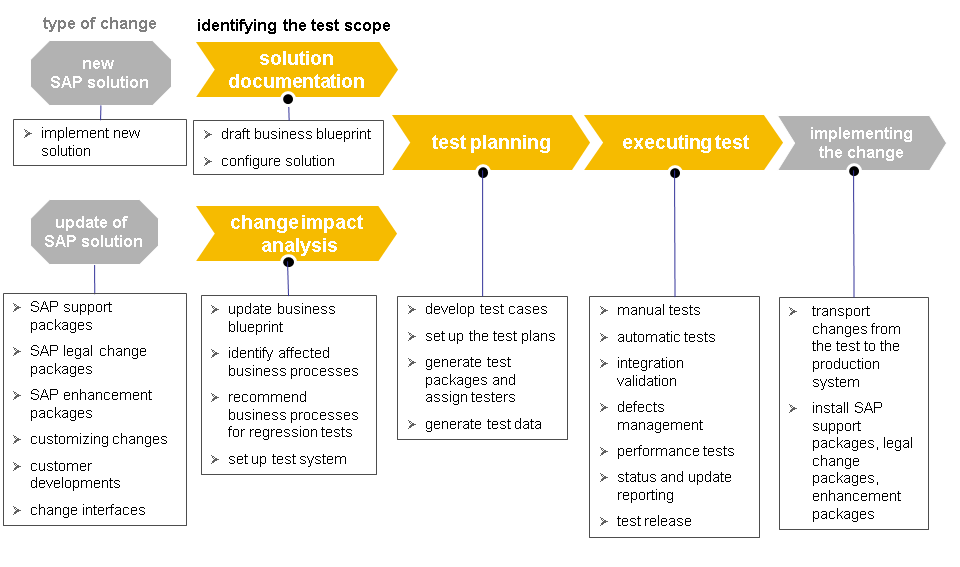
In detail, these areas of the Solution Documentation model provides the following features:

* **Developments and Executables Library**
  + Component-oriented container of technical objects and their documentation, e.g. Web Dynpro applications, transactions, custom reports etc.
  + Technical objects can be reused in different process steps and end-to-end processes
  + Libraries can be partially automatically generated based on usage statistics and structured by the application component hierarchy (ACH)
* **Process Step and Interface Library**
  + Function-oriented container of all process steps and their business-context independent documentation
  + Process steps can be reused in different end-to-end processes
  + Automated, on-demand generation of the library structure based on the ACH, which can be manually extended
* **Configuration Library**
  + Function-oriented container of configuration activities, grouped into reusable units
  + Configuration units can be reused for repeated configuration tasks
* **End to End Processes**
  + Business-oriented documentation of processes describing your end-to-end scenarios
  + Library elements, e.g. process steps, interfaces and technical objects, can simply be reused

The library concept enables you to re-use and to reference all the objects mentioned above whenever possible. This is the key to avoid redundant documentation.



**Test Management Phases:**



## **Integration**

Before you start testing, you must have created a project structure with your business processes in the solution documentation and configured your business processes. For more information, see [solution documentation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3ef70f5452e7024be10000000a44176d.html).

You can use the testing tools of SAP as well as solutions from partners, with the following combinations:

* [SAP Solution Manager test suite](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/c55b955524f44672b52fb77afc90f1f8.html) and the Test Data Migration Server (TDMS, see [https://support.sap.com/solution-manager/integrated-tools/tdms.html[Information published on SAP site](http://help.sap.com/disclaimer?site=https://support.sap.com/solution-manager/integrated-tools/tdms.html)](http://help.sap.com/disclaimer?site=https://support.sap.com/solution-manager/integrated-tools/tdms.html))
* SAP Solution Manager, the SAP TDMS,

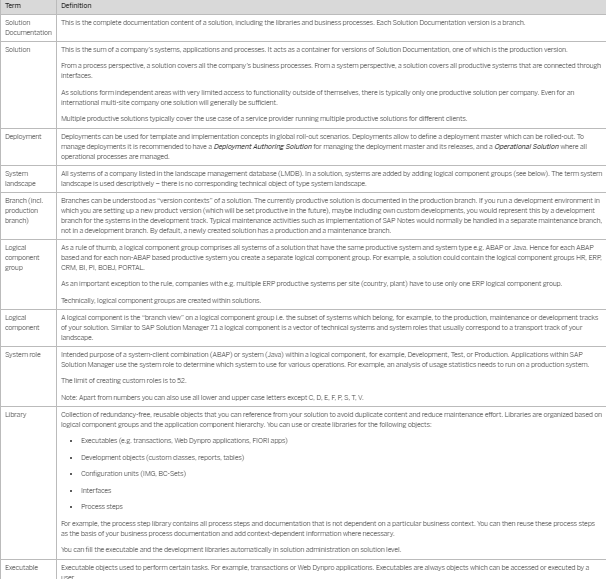
[SAP Quality Center by HP](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/146c6c5743289627e10000000a441470.html), and SAP Test Acceleration and Optimization (see [http://help.sap.com/saptao30[Information published on SAP site](http://help.sap.com/disclaimer?site=http://help.sap.com/saptao30)](http://help.sap.com/disclaimer?site=http://help.sap.com/saptao30))

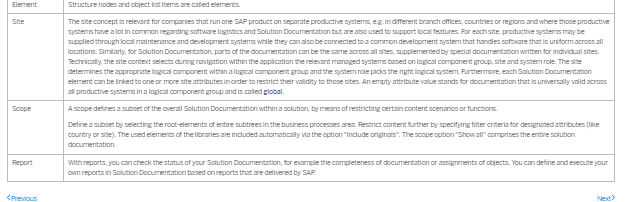
* SAP Solution Manager, the SAP TDMS, the SAP Quality Centre and other tools for partner test management.

**SAP Solution Manager**

SAP Solution Manager provides various tools and methods for implementing and maintaining both SAP and customer solutions. The Solution Documentation model is based on a hierarchical structure, which uses a solution as the single, central point of access for all Solution Documentation content.

**Basic Terms :**





**Libraries**

* Process Step Library

The process step library is a business oriented collection of re-usable process step originals used to assemble processes. It is grouped by functional domains that correspond to organizational structures (for example financial, production or sales). All process step originals should exist only once to obtain an overlap-free library structure.

* Executable Library

The executable library is a functional oriented collection of re-usable executable originals used to facilitate process step originals with execution means. It is calculated using managed system usage data (performance database and usage and procedure logging). All executable originals are automatically grouped by logical component groups on the first level and on the following using the application software structure (application component hierarchy or development package).

* Development Library

The development library is a functional oriented collection of re-usable development object originals to document executable customer enhancements. The development object originals are automatically grouped by logical component groups on the first level and on the following using the applications software structure (application component hierarchy or development package).

* Interface Library

The interface library is a functional oriented collection of re-usable interface originals used to depict how system breaks are bridged. It is organized according to scenario and functional aspects and follows operational needs.

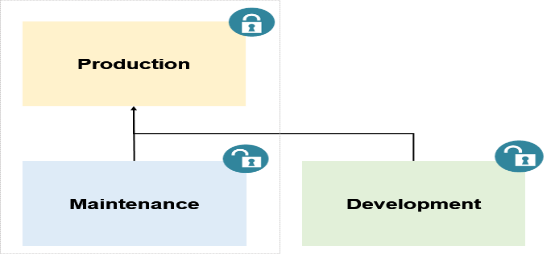
* Configuration Library

The configuration library is a functional oriented collection of re-usable configuration units used to describe the configuration of single functions, complete applications, or processes. It is grouped by functional domains that correspond to organizational structures (for example financial, production or sales) and organized according to a process organization.

Lifecycle based on Branches

A branch represents a version of the Solution Documentation containing processes, libraries, and systems.

* The production branch represents the productive version of the entire Solution Documentation.
* The maintenance branch represents the editable version of the productive Solution Documentation. It provides an safe environment for performing changes.
* The development branch represents the development version for future Solution Documentation.



Relationship of Branches

There is always a production and a maintenance branch but customers can define as many additional branches as required. A child branch, like a maintenance branch, has always full visibility into the Solution Documentation of the parent, for example in the maintenance branch, all unchanged elements have the state of the production branch.

The branch setup should be driven by the customers system tracks and the planned customer releases.

## **Prerequisites**

Configuration of Process Management in SAP Solution Manager Configuration

In SAP Solution Manager Configuration, you have to do the following:

* You have to configure all applications-relevant sub scenarios in Cross Scenario Configuration.
* You have to configure all sub scenarios in Mandatory Configuration.

For a correct graphical representation of solutions, you have to configure the scenario Infrastructure Preparation, step Enable Gateway Services. In the list of gateway services for Process Management you have to activate the services AGS\_GBC\_ODATA\_BPMN\_SRV (Graphical Component BPMN Gateway Service), AGS\_GBC\_ODATA\_GOM\_SRV (GBC GOM Gateway Service), and PROCESSMANAGEMENT (SAP Solution Manager APIs for third party tool integration).

* You have to configure the scenario Embedded Search in order to establish the connection between Embedded Search and TREX or SAP HANA.
* You have to configure the process management in the scenario Process Management.

**Functional Highlights of the Solution Documentation:**

Usability

You use a modern browser-based user interface making the administration and documentation of SAP solutions simple, intuitive and business-focused, and you benefit from graphical process modeling.

Re-Use

You can organize reusable documentation for business process steps and other entities such as transactions, development objects, and configuration activities in libraries. Reduce your content management effort by working on reusable documentation elements only once for multiple use in different end to end business processes.

Integration

You can take advantage of the concept of Solution Documentation as a single source of truth for all content related to business processes. A solitary solution will support all application lifecycle phases from implementation through to operations. Do away with creating solutions altogether when they are not needed, for example for sole use of basic SAP services, such as EarlyWatch Alert.

Versioning

You can simplify maintenance, new development and upgrades by using the version concept – branches. With this concept, you are part of the quantum leap in SAP release management.

* Big Solutions

You can handle solutions of all shapes and sizes with multi-level hierarchies without limitations on the size and hierarchical complexity.

* Custom Extensions

You can add extra structure levels and your own attributes by taking advantage of the flexible, model-based architecture of Solution Manager 7.2 to systematically extend SAP standards.

How Solution Documentation works

The Solution Documentation model is based on a hierarchical structure, which uses a unified solution concept as the single, central point of access for all Solution Documentation content. The solution comprises the description of your system landscape as well as the documentation of your business processes, interfaces, technical objects and your custom developments.

Thus, the solution concept stands in the center of the Solution Documentation. It is the sum of a company’s system landscape, applications and processes. It acts as a container for versions of Solution Documentation, one of which is the production version. The terms systems landscape, process and version have the following meaning:

* System Landscape

Logical Component Groups are groups of Logical Components. Logical components point to actual technical systems.

* Process

A process is a set of logically related activities and executes functions in systems to achieve a business goal.

* Version

Each Solution Documentation version is a branch containing processes, libraries, and systems.

You organize your Solution Documentation into the following different areas:

* Libraries of reusable documentation for technical objects, such as development objects, executables, and configuration activities
* Library of business process steps where you can reuse the documentation from the other libraries
* Library of interfaces where you document the interfaces used by your end to end processes
* End to End Business Processes where you model your scenarios and processes by reusing the process steps, interfaces, and technical objects from the libraries,

In detail, these areas of the Solution Documentation model provides the following features:

* Developments and Executables Library
  + Component-oriented container of technical objects and their documentation, e.g. Web Dynpro applications, transactions, custom reports etc.
  + Technical objects can be reused in different process steps and end-to-end processes
  + Libraries can be partially automatically generated based on usage statistics and structured by the application component hierarchy (ACH)
* Process Step and Interface Library
  + Function-oriented container of all process steps and their business-context independent documentation
  + Process steps can be reused in different end-to-end processes
  + Automated, on-demand generation of the library structure based on the ACH, which can be manually extended
* Configuration Library
  + Function-oriented container of configuration activities, grouped into reusable units
  + Configuration units can be reused for repeated configuration tasks
* End to End Processes
  + Business-oriented documentation of processes describing your end-to-end scenarios
  + Library elements, e.g. process steps, interfaces and technical objects, can simply be reused

# Test Suite Launchpad Group

# [Launchpad](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/53079555252e292de10000000a44147b.html) group is the central access point for all the information and functions you need to determine the scope of testing required for cross-system processes, manage the tests centrally, and execute the tests.

## **Features**

The group consists of the following tiles displayed by default:

| **Tile** | **Description** |
| --- | --- |
| **Test Suite - Overview** | You can review status overviews on the following topics, which are distributed on tabs:  ● **Test Planning** tab: Test plans and the assigned test cases and test packages in the whole system, with information such as dates, responsible persons, number of tests, test case status, and defects.  ● **Test Execution** tab: The user's own test plans and the assigned test cases and test packages, with the same information as under Test Management.  ● **Business Process Change Analyzer** tab: number of changes found for a system, caused by support packages, enhancement packages, business functions, or transports. |
| **Test Suite - Test Preparation** | You can review your solution hierarchy. Navigate to the included e xecutables, test cases, related documents (such as business requirements and incidents) and testing information. From here, you can execute completeness and gap reports for the solution. |
| **Partner Test Management** | You can manage the associations of your solution in SAP Solution Manager with the test projects of partner tools. |
| **My Tasks - TBOM Worklist** | You can display and edit TBOM recording tasks as business expert or tester. TBOMs are required for the Business Process Change Analyzer to perform change analyses. For more information, see [My Tasks: TBOM Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/753ed082dc794602b5a7b8781e5b2230.html). |
| **Business Process Change Analyzer** | You can changes the system, for example support packages, customer developments or add-on installations that can affect different business processes. To ensure that the business processes still work correctly after such a change, identify and test the affected business processes or parts thereof. For more information, see [Business Process Change Analyzer (BPCA)](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/e7ae57e0291149adaca93a9dcce33abb.html). |
| **Scope and Effort Analyzer - Upgrade Planning** | You can estimate the development and test efforts for planned maintenance projects to implement enhancement packages or support packages. The application shows you an overview of analyses and provides several options on how to process analyses. For more information, see [Scope and Effort Analyzer](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/f5e56c52f01e7554e10000000a44176d.html). |
| **Test Plan Management** | As a test organizer, you can use this view to manage test plans and test packages, for example, assign testers to test packages and define attachments. For more information, see [Test Plan Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/48e151fddb163184e10000000a421937.html). |
| **Review Tester Worklists** | You can access all test cases assigned to a specific tester. For more information, see [Review Tester Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/6b45b3d03bcc407fb48ba2332dafc992.html). |
| **My Tasks - Tester Worklist** | You can access all test cases assigned to you as a tester.For more information, see [My Tasks - Tester Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/81d7dcadebb24422bde83220928684a4.html). |
| **My Tasks - Damaged Test Cases** | You can access all damaged test cases. For more information, see[Damaged Test Cases](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/8ba28ac59e8d44deaae6fb9fd9980001.html)  . |
| **Test Execution - Log** | You can display the execution logs of automated test cases.For more information, see[Test Execution Reports](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4736e055791db61ae10000000a44538d.html). |
| **Test Execution - Jobs** | You can display information about the background jobs with which you scheduled automatic test cases:  ●**Job Status**: Did the job run?  ●**Test Status**: What was the result?  ● **Log ID**: ID of the execution log for the job to which a test package is assigned.  ●**Job Name**: Link to job execution details  For more information, see [Test Execution Reports](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4736e055791db61ae10000000a44538d.html). |
| **Test Repository - Test Configurations** | You can access the test composition environment through the test repository, where you can create and edit automated test cases. For more information, see [Test Repository: Test Scripts and Configurations](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b5e6e0ffd8474be08453a43b310f7cc3.html). |
| **Test Repository - Test Scripts** | You can access individual test scripts through the test repository, which are part of the test configurations. For more information, see [Test Repository: Test Scripts and Configurations](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b5e6e0ffd8474be08453a43b310f7cc3.html). |
| **Test Suite - Analytics** | You can see different reports related to test management. For more information, see [Test Suite - Analytics](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/c237def1326e4d7282face0222f40391.html)  . |
| **Personalization - Test Suite** | You can access personalization application to define your preference settings. For more information, see [Personalization](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a72befe812c44f6ba9e6804262bc4ba4.html). |

The following tiles are not displayed by default:

| **Tile** | **Description** |
| --- | --- |
| **Configuration - Test Suite** | You can access Solution Manager configuration for test suite scenarios. For more information, see [Configuration of the Test Suite](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/572ede554cf5d207e10000000a44538d.html). |
| **Administration - Test Suite** | You can access basic settings for the business process change analyzer (BPCA). For more information, see [Administration - Test Suite](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/cf60aa80c5c84698bc794168c392b5f2.html). |
| **Administration - Change Impact Analysis** | You can access basic settings for the test suite. For more information, see [Administration - Test Suite](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/cf60aa80c5c84698bc794168c392b5f2.html). |

# SAP Solution Manager Test Suite

# Use

# The test functions and tools are almost all provided by SAP Solution Manager and SAP Test Data Migration Server. The following graphic illustrates the process.

# Fig: SAP Solution Manager and Test Automation Framework

# https://help.sap.com/doc/PRODUCTION/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/loiof295e8dac3984230976ef37086ca3e7d_LowRes.png

## **Process**

1. Identify the test scope.

Perform the following process steps, depending on the change type:

* + **New SAP solution**: The SAP Solution Manager [Solution Documentation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3ef70f5452e7024be10000000a44176d.html) functions design and document business processes, and catalog them hierarchically. The user departments formulate business requirements in the solution documentation. They assign manual and automatic test cases to the business processes in the solution documentation.

Create automatic test cases based on[Test Automation with eCATT](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4c4dbc20f87e4108983c16774438364e.html), [Component-Based Test Automation (CBTA)](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/99ed237d141240cc839abc6c9efbd6a4.html), or certified external tools, directly in the business process context, with the [Test Automation Framework](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/02cbe8897138401f8f04ec4627d111de.html). You can manage your test cases in the [Test Repository](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b5e6e0ffd8474be08453a43b310f7cc3.html).

* + **Update an existing SAP solution**: Check whether important applications or critical business processes are affected by the update, with the SAP Solution Manager [Business Process Change Analyzer](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/e7ae57e0291149adaca93a9dcce33abb.html).

1. Plan the tests and set up test systems.
   * The SAP Test Data Migration Server (TDMS) supports the automatic creation of test systems with low data volume. (For more information, see [https://support.sap.com/solution-manager/integrated-tools/tdms.html[Information published on SAP site](http://help.sap.com/disclaimer?site=https://support.sap.com/solution-manager/integrated-tools/tdms.html)](http://help.sap.com/disclaimer?site=https://support.sap.com/solution-manager/integrated-tools/tdms.html).
   * Create one of the following types of test plans:
     + In the [Test Plan Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/48e151fddb163184e10000000a421937.html) application, you can display, change, copy or delete test plans.

In the **Business Process Change Analyzer** application, you can analyze business processes that are affected by a change and have to be tested again. Base your test plans on the analysis results.

* + Summarize selections of test cases to test packages and assign the test packages to testers.
  + Schedule automatic tests, for example, each night.
  + For better test execution control, base a workflow on a [Release Status Schema](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/5ab1cae0f3944d5dba6cc2b4c1a25441.html). Testers will receive an e-mail when they can start the execution of their assigned test packages, or when the execution must be stopped.

1. Perform the test.

The **Test Suite** group in the SAP Solution Manager launchpad is the central point of entry for performing and evaluating tests.

* + In the [My Tasks - Tester Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/81d7dcadebb24422bde83220928684a4.html) application, the tester sees a list of all test packages assigned to him or her.
  + The tester can also call and process his work list directly via the link in the e-mail notification.
  + With [Test Suite - Analytics](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/c237def1326e4d7282face0222f40391.html), you can monitor the status of your test cases and executions. The BI reporting functions display graphically formatted overviews of the status of the test plans or the processing history.

# Test Suite in SAP Solution Manager

## Use

The **Test Suite** group in the SAP Solution Manager launchpad is the central access to all functions for preparing, creating, managing, and evaluating test plans, test scripts, test configurations etc.

Furthermore, you can use the [business process change analyzer (BPCA)](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/e7ae57e0291149adaca93a9dcce33abb.html)to perform change influence analyses, and generate tailored test plans from the results, and the scope and effort analyzer to calculate the efforts to test and install new software.

## Features

From the **Test Suite** group in the SAP Solution Manager launchpad, you can start the following views:

**Test Suite - Overview**

This application gives you status overviews on the following topics, which are distributed on tabs:

* **Test Planning** tab: Test plans and the assigned test cases and test packages in the whole system, with information such as dates, responsible persons, number of tests, test case status, and defects.
* **Test Execution** tab: The user's own test plans and the assigned test cases and test packages, with the same information as under **Test Management**.
* **Business Process Change Analyzer** tab: number of changes found for a system, caused by support packages, enhancement packages, business functions, or transports.

**Test Suite - Test Preparation**

Overview of your solution hierarchy. You can navigate to the included executables, test cases, related documents (such as business requirements and incidents) and testing information. From here, you can execute completeness and gap reports for the solution.

**Partner Test Management**

In this application, you can manage the associations of your solution in SAP Solution Manager with the test projects of partner tools.

**My Tasks - TBOM Worklist**

Display and edit TBOM recording tasks as business expert or tester. TBOMs are required for the Business Process Change Analyzer to perform change analyses. For more information, see [TBOM Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/753ed082dc794602b5a7b8781e5b2230.html).

**Business Process Change Analyzer**

Changes to the system, for example support packages, customer developments or add-on installations, can affect different business processes. To ensure that the business processes still work correctly after such a change, identify and test the affected business processes or parts thereof. For more information, see [Business Process Change Analyzer](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/e7ae57e0291149adaca93a9dcce33abb.html).

**Scope and Effort Analyzer**

Estimates the development and test efforts for planned maintenance projects to implement enhancement packages or support packages. For more information, see [Scope and Effort Analyzer](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/f5e56c52f01e7554e10000000a44176d.html).

**Test Plan Management**

As a test organizer, use this view to manage test plans and test packages, for example, assign testers to test packages and define attachments. For more information, see [Test Plan Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/48e151fddb163184e10000000a421937.html).

**Review Tester Worklists**

Access all test cases assigned to a specific tester. For more information, see [Review Tester Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/6b45b3d03bcc407fb48ba2332dafc992.html).

**My Tasks - Tester Worklist**

Access all test cases assigned to you as a tester. For more information, see [Tester Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/81d7dcadebb24422bde83220928684a4.html).

**My Tasks - Damaged Test Cases**

Access all damaged test cases. For more information, see [My Tasks - Tester Worklist](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/81d7dcadebb24422bde83220928684a4.html).

**Test Execution - Log**

Display the execution logs of automated test cases.

For more information, see [Executions](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4736e055791db61ae10000000a44538d.html).

**Test Execution - Jobs**

Display information about the background jobs with which you scheduled automatic test cases:

* **Job Status**: Did the job run?
* **Test Status**: What was the result?
* **Log ID**: ID of the execution log for the job to which a test package is assigned.
* **Job Name**: Link to job execution details

For more information, see [Executions](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4736e055791db61ae10000000a44538d.html).

**Test Repository - Test Configurations**

The test repository gives you access to the test composition environment, where you can create and edit automated test cases. For more information, see [Test Repository](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b5e6e0ffd8474be08453a43b310f7cc3.html).

**Test Repository - Test Scripts**

The test repository gives you access to individual test scripts, which are part of the test configurations. For more information, see [Test Repository](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b5e6e0ffd8474be08453a43b310f7cc3.html).

**Test Suite - Analytics**

Different reports related to test management. For more information, see [Analytics](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/c237def1326e4d7282face0222f40391.html).

**Personalization - Test Suite**

Access personalization application to define your preference settings. For more information, see [Personalization](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a72befe812c44f6ba9e6804262bc4ba4.html).

**Configuration - Test Suite**

Access Solution Manager setup for test suite scenarios. For more information, see [Configuration of the Test Suite](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/572ede554cf5d207e10000000a44538d.html).

**Administration - Change Impact Analysis**

Basic settings for the business process change analycer (BPCA). For more information, see [Administration](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/cf60aa80c5c84698bc794168c392b5f2.html).

**Administration - Test Suite**

Basic settings for the test suite. For more information, see [Administration](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/cf60aa80c5c84698bc794168c392b5f2.html).

# Test Automation with the Test Automation Framework:

## **Use**

The Test Automation Framework is integrated in the SAP Solution Manager system landscape and provides functions such as creating test configurations and test scripts for automated tests, test data containers and system data containers.

## **Features**

* Test preparation
  + Set up automated tests in the **Test Suite - Test Preparation** application. For more information, see [Solution Documentation: Creating Test Cases](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/df49e0555937e263e10000000a44538d.html).
  + When you define the landscape of your solution, the corresponding system data container is automatically created or updated with the relevant logical component group.
  + The SUT Management UI (SUT = system under test) allows to enhance system data container (SDC) information, which is automatically derived from the solution documentation. When test cases are executed, the RFC destinations are derived automatically depending on the logical component and system role.
  + When creating a test data container, you create parameters. The Test Automation Framework supports plausibility checks for parameters even of those data elements which are not used in SAP Solution Manager but only in the target system.
  + When editing the content of an eCATT test script , the interface of transactions or function modules is imported automatically from the SAP Solution Manager system landscape and displayed, if a system data container is assigned to the test case.
* **Test plan management:**

Assign the system role to the test plan. The RFC destinations are derived automatically when the test cases are executed

# Partner Test Management

## **Use**

Partner test management integrates SAP Solution Manager and the partner's test management application. This allows the transfer of business blueprints, related documents, specifications, business requirements, links, and test objects such as transactions and custom programs. It can help you to plan an efficient testing phase.

This function is not available by default. To be able to use it, you must purchase SAP Solution Manager partner test management, and activate the business function (ST\_QCA) explicitly. For more information, see [2433483 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/2433483)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/2433483).

You can open the **Partner Test Management** application on the SAP Solution Manager launch pad.

## **Features**

In the **Branch Association** section, you can do the following:

* Create, change and delete associations: Associations refer to a one-to-one mapping of a partner project to a combination of branch and one or multiple scopes.

Note that you cannot assign a single partner project to several SAP solutions in one SAP Solution Manager system.

When an association is created, you can edit the scopes of the association.

You can delete an existing association. Then it is free to be associated with a different branch.

* **Push Solution Documentation**: You can push the solution documentation from SAP Solution Manager to the partner test management system. Only elements of type Executables, Documents, and Process Variants are trasferred. After this, the quality manager can perform test-related activities, such as creating test cases and assigning defects to the test cases.
* **Display Test Results**: After pushing the solution documentation to the partner system, you can view the test results by choosing **Display Test Results**. It includes the following details, for example:
  + The status of the test cases. For example, if there is an error, you can view the number of unresolved defects (in the SAP system) that are associated to the test case.
  + The time at which the solution document was last synchronized and the user who synchronized it.
  + If you select the **Display Nodes with Test Results** checkbox, you can view only those nodes that have test results.
  + You can search for the business process for which you want to view the test results by using the filter option in **Business Process**.
* **Refresh Solution Documentation**: If you select the check box, any changes made to the solution documentation are automatically updated in the partner test management system. There is a background job that runs daily to ensure that the updated information is available in the partner system.
* **Refresh Test Results**: If you select the check box, the updated test results in the partner test management system are transferred to SAP Solution Manager. An hourly job runs in the background to ensure that the updated test results are available in the SAP Solution Manger system.
* **Push Status**: Shows the status of the solution documentation. It reflects the status when the solution documentation is pushed from SAP Solution Manager to the partner system, and also when the solution documentation is pulled from SAP Solution Manager by the partner system.

In the **Details** sections, you can do the following:

* You can filter document types and executable types that are transferred from SAP Solution Manager to the partner system.
* **Key Word Mappings**: You can map custom attributes in SAP Solution Manager by using the user-defined fields of the partner system.
* **Logs**: You can view details, such as the time at which the solution documentation was pushed from SAP Solution Manager to the partner system, and details about the user who has performed the action.

# Partner Test Management Configuration

## Use

To be able to use the test suite with partner tools, you have to set it up in SAP Solution Manager.

## Process

In **SAP Solution Manager Configuration** (transaction SOLMAN\_SETUP), go to **Test Suite**  **SAP Quality Center/SAP TAO**  **Partner Test Management.**

Follow the guided procedure. For more information, see the help texts in each step and activity.

# Infrastructure and Test Tools

The following sections describe the different tools and user interfaces that are used for test management with SAP Solution Manager 7.2:

* [Test Suite in SAP Solution Manager](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/8396c90f3c53406182ba2d884185f1b3.html)
* [SUT Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/69c71656795f5f5ae10000000a4450e5.html)
* [Creating Test Configurations in the Test Automation Framework](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/02cbe8897138401f8f04ec4627d111de.html)
* [Test Automation with eCATT](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4c4dbc20f87e4108983c16774438364e.html)
* [Component-Based Test Automation (CBTA)](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/99ed237d141240cc839abc6c9efbd6a4.html)

**Component-Based Test Automation (CBTA)**

**Use**

Component-based test automation (CBTA) allows you to create automatic tests.

An automatic test is made up of a test script that includes components. There are two kinds of components: default and screen components:

* **Default components** perform basic actions like clicking on a button or selecting a tab. They are delivered in the SAP Solution Manager system.
* **Screen components** are generated for either SAP GUI transaction screens or SAP CRM Web UI views. Web Dynpro ABAP application views are supported, but only in native mode, not when embedded in the portal. They are created with a parameter for each field on the screen. The value of a parameter is given to the corresponding field during execution. Screen components are generated automatically at the end of a test recording.

There is only one screen component per SAP GUI transaction screen, SAP CRM Web UI application view, and SAP Web Dynpro application. And there is only one screen component per logical component to ensure that the tested application is of a dedicated version. So a screen component can be shared among several tests.

Use the **Solution Documentation** application to create and maintain test configurations. The test scripts created with the CBTA tool are compatible with the eCATT framework.

Note

You can test business processes of SAP Products. The following UI types are supported: SAP UI5 Fiori, SAP GUI, SAP CRM Web UI, SAP FPM, SAP Web Dynpro ABAP, SAP Web Dynpro Java, and BSP.

**Integration**

* SAP Solution Manager: Store data required to create, optimize, and maintain tests and components.
* Test composition environment: Create test configurations and test scripts recorded with CBTA. For more information see Test Repository
* Test plan management: Schedule the execution of tests.

# Test Repository: Test Scripts and Configurations

# The applications Test Repository - Test Configurations and Test Repository - Test Scripts give you access to the test composition environment in SAP Solution Manager. Here, you can create and edit test scripts and test configurations for automated test cases:

* A test configuration contains a system data container, a test data container, and one test script. The data from these elements can be reused in several test configurations.
* A test script describes in technical detail the components and sequence of activities that are to be tested, for a selected business process. The test script can contain one or several steps, which you can put into the right order.

You can create test scripts automatically by using test tools that are configured in the system, either eCATT, component-based test automation (CBTA), or third-party.

You can reuse existing test scripts, create modular test scripts and test end-to-end processes.

* A test case is the assignment of a test configuration to a structure node (for example a business process step) in your solution documentation.

**Features**

* Recording tests scripts ( SAP S/4 HANA support improvements)
* Quick repair: Continue previous recording.
* Cross technology recording in one session (ex: start with CRM and continue on SAP UI5...)
* Record multiple SAP SUI windows
* Record embedded HTML into SAP GUI transaction
* Record IE: SAP GUI continues recording
* Include one CBTA script into another CBTA script (modularized some action like a test step )
* Easy maintenance ( default/screen component concept, mass update of control identifier in all test scritps,…)
* Managing test configurations with shared components.
* Managing screen components and their parameters.
* Using the test data container (TDC) to define several sets of data.

**A CBTA test is an eCATT test so it can use the eCATT features. The object structure is defined as follows**:

* The solution documentation references test configurations to test selected executables.
* The **Test Repository - Test Scripts** and **Test Repository - Test Configurations** applications lists all existing test configurations and test scripts.
* A test configuration includes the following assets:
  + A system data container (SDC) to define on which system the test recording and execution are performed. There is only one SDC attached to a solution. In this scenario, the SDC is automatically generated with the same name as the solution, prefixed with the character Z.
  + A test data container to define different sets of data to provide the input parameters for the tests.
  + A test script which includes the following:
    - The executable object that defines the transaction, program, or URL for the test. It is linked to a logical component group, as defined in the related solution landscape.
    - A test profile that defines which business user is to be used for recording and execution. It is defined in the **SUT Management** application.

The steps of the automatic tests, for example, clicking a button, selecting a tab, or entering data in an input field.

# Installing and Configuring CBTA

## Use

Installing and configuring component-based test automation (CBTA) comprises the following activities:

## Activities

1. Define and configure the systems under test (SUT), and create users for them.
2. To allow CBTA access to the SUTs, define RFC or HTTP destinations and specify the user credentials.
3. Download the CBTA client and install it locally.
4. To ensure the installation and configuration of the CBTA client are correct, perform self-checks.

# Configuring CBTA in SAP Solution Manager

## Use

Configure the systems under test (SUTs) and create users. To allow, for example, CBTA to record and execute test scripts on SAP Solution Manager SUTs, maintain a business user that the CBTA client can use.

## Procedure

1. In SAP Solution Manager Configuration (transaction SOLMAN\_SETUP), choose Test Suite SAP Solution Manager Test Automation.
2. In SAP Solution Manager Configuration (transaction SOLMAN\_SETUP), choose Test Suite SAP Solution Manager Component-Based Test Automation.
3. Follow the guided procedure.

# Creating Test Profiles and Logon Data for SUTs

## Use

In the system data container (SDC), you define the access information for the systems under test (SUT). These are the systems in which tests are recorded and executed. This requires RFC and/or URL, user names, and passwords.

In the test composition environment (which is accessible when you edit a test configuration in the Test Repository - Test Configurationsapplication), the test profile associates a user and password with a test script. It allows you to reuse combinations of destinations and users for different system roles. When you change the system role, depending on the type of definition, the credentials can be role-dependent or role-independent.

Basically, a test profile is a name that is associated to a user name and password. In the editor of the test repository, a test profile is assigned to a test script.

In the SUT Management application you can define test profiles. They can be defined via logical component group, site and system role. In case of component-based test configuration (CBTA), following the UI technology recorded the test profiles are defined along with a technical RFC destination to retrieve technical information from the SUT or simple URL.

Administrating Access Authorizations

See SUT Management security aspects.

Access Procedure

* From WorkCenter:

In the Solution Manager WorkCenter go to the Test Suite section and click on the Tile Administration. Then in the Administration application of Test Suite select the tab Test Automation Framework and click on SUT

* From Test Composition Environment:

Open a test in Test Composition Environment. In the toolbar at the top click on Goto button then select SUT Management in the drop down list.

# Using SUT Management

SUT Management uses System Data Container (SDC) to retrieve System Under Test (SUT) information and enhance them with Test Profiles.

Import SDC

To be able to create a Test Profile for a SUT the SDC should be imported in SUT Management. Once imported, the SDC is visible in the enhanced SDC hierarchy.

Note

The SDC name is the concatenation of “Z” with the solution technical name.

The SDC node contains the Logical Component Group (LCG) defined in the solution linked with the SDC. Under a LCG node there are the Logical Components (LC) matching the solution branches and sites. Under the LC node there are the system roles which point to the target system.

Hierarchy Structure

SDC (Solution)

* LCG1 (CRM)
* LC1 (Branch Production)
  + System role1 (Production)
  + System role2 (Quality)
  + System role3 (Development)
* LC2 (Branch Maintenance – Site France)
* LC3 (Branch Maintenance – Site England)
* LCG2 (ERP)

Synchronize SDC

While changing the SDC structure by adding or removing LCG, LC or system role from the solution, SUT Management should be synchronized so you can work with the up to date SDC structure.

# Configuring a SUT With Test Profile

To setup a SUT, select a system role in the hierarchy.

The Test Profile name is used as an attribute of a test script to represent the business user in different systems. It means that the Test Profile name can be the same in different system roles of a LC so there is no need to have the same user and password for each system in the landscape.

The Propagate button replicates the Test Profile along with its user and password through the whole LCG.

SUT Management distinguishes two kinds of SUT. Systems which are SAP ABAP backend and systems based on URI.

1. **SAP ABAP Backend**

For SAP ABAP backend the setup is different depending on the tool used for the test. CBTA needs a technical destination and then you can add Test Profiles. All other tools are based on RFC destinations with login user.

**CBTA Technical destination for SAP ABAP backend**

CBTA needs a technical destination which should be created with the CBTA Configuration scenario.

The destination can automatically be retrieved using the button **Propose Destination**. It uses the destinations created by LMDB for the selected system to retrieve the system information and compute the technical destination name if it exists.

The destination can be checked via the dedicated button. As a result, the destination status for the selected system is updated.

**CBTA Test Profile for SAP ABAP backend**

For CBTA, a Test Profile consists of a Test Profile name, a business user and its password on the SUT. The user existence can be checked via the**Check** button. As a result, the status will be updated.

**eCATT and third party tools SAP ABAP backend**

For eCATT and third party tools, a Test Profile consists of a Test Profile name and a destination. The destination is used by the tools to logon to the system so it contains the business user.

1. **URI based SUT**

URI based SUT are systems which cannot be reached by RFC from Solution Manager.

The Test Profile consists of a Test Profile name, a connection type, a connection URI, a business user, and a password. There are three types of Test Profiles which are identified by their “Connection Type”.

The Base URL and the HTTP destination connection types are used with the executable object of type URL.

The External Service type is used while the system is of type external service (Cloud).

**Base URL**

The executable object value of type URL is linked to the connection URI. The connection should result in a valid URL (protocol, host, port at least).

**HTTP Destination**

The connection URI value is the name of the HTTP destination. The protocol, the host and the port are determined from the HTTP destination. Then the executable object of type URL is linked to it.

**External Service**

The connection URI value is automatically retrieved from the system of type external service. The connection URI and the executable object (all supported types but transaction) are computed to generate a valid URL.

# Related Test Scripts

Executing a test script for systems with different roles

You can execute a test script for systems with different roles if, for example, you have an SDC target component that contains the development system and the quality assurance system of your SUTs.

Prerequisite: The same test data is available on both SUT instances, in the development and the quality assurance systems.

1. Create test profiles with the same name but with the different roles of your SUT.
2. Create a CBTA test script on the development system.
3. Change the system role. Without changing the script, you have switched the target system on which the test is to be executed

# Installing and Configuring the CBTA Client

## Context

Download the CBTA client and install locally. CBTA allows you to configure the following:

* On request by SAP support, you can adapt the log settings and download error logs.
* To improve readability, you can adjust the theme of the CBTA user interface in settings.

First Installation

If a previous version of CBTA using the MSI is installed, it is recommended to uninstall it manually before installing the new version of CBTA using the SAP Setup Installer.

Note

In addition, it is recommended to close all SAP GUI windows and ensure that the SAP Logon process is not running before installing or uninstalling CBTA.

## Procedure

1. Download the CBTA client and install it locally. The CBTA installation package can be found here:
   1. Call https://support.sap.com/software/patches.html
   2. Choose A-Z (Alphabetical List of Products) and navigate to SAP Solution Manager  SAP Solution Manager 7.2.
   3. Choose Entry by Component  Support user Frontend GUIs  COMP. BASED TEST AUTOM. 3.0  Win 32.
   4. Choose the ZIP file for the latest patch.

CBTA 3.0 SP05 is delivered in a SAP Setup installer, like SAP GUI. This installer offers more features than the Microsoft System Installer (MSI), used for previous releases of CBTA.

For example, with SAP Setup it is possible to:

* Maintain nearly all Windows™-based SAP front end components on a single installation server
* Deploy SAP components on workstations without requiring administrative privileges on all machines
* Automatically update the workstations and reboot them if necessary with the Automatic Workstation Update Service, whenever the installation server is patched, or the packages installed are updated.

The new CBTA installer is an auto-extractable file, which will run the SAP front end installer when executed.

SAP Setup Guide

The SAP Setup Guide is included in the CBTA installer: the content of the auto-extractable file can be extracted using WinRAR, the SAP Setup Guide.pdf file is in it. This document describes all the features and capabilities of SAP Setup and explains how to setup an installation server.

| SAP Setup Latest Information | |
| --- | --- |
|  | |
| **SAP Note** | **Description** |
| [1587566 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1587566)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1587566) | Detailed release notes and known issues of SapSetup version 9.0. Description how to retrieve the latest version of SAP Setup. |
| [1583967 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1583967)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1583967) | General release information note for SAP Setup 9.0 |
| [1162270 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1162270)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1162270) | Installing the DS installation service (analysis notes). Additional information on how to create and maintain an installation server to distribute the SAP Frontend Software for Windows to client computers using Local Security Handling. |
| [1177282 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1177282)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1177282) | End of support for SAP releases using Microsoft products. |

Configure CBTA Client

1. To start the client configuration, do one of the following:
   * Under Microsoft Windows, choose Start  CBTA  CBTA Configuration.
   * In the Administration - Test Suite view, choose CBTA Maintain Client Settings.
   * Under Microsoft Windows, choose Start  CBTA  CBTA Configuration.
2. You have the following options:
   * On the Recording tab page, you have the following options:
     + Under Component Creation you can record test scripts without screen components.

Compose a test script based only on default components.

* + On the Log tab page, adapt the log settings if required by SAP support.

1. Save your entries.

# Performing Self-Checks

## Prerequisites

* You have installed the CBTA backend componentsYou have installed the CBTA client locally.

## Context

With the self-checks for component-based test automation (CBTA), the administrator of the SAP Solution Manager system can verify the installation and configuration of the CBTA client and system under test (SUT) accesses. Self-checks are also useful for SAP support to identify issues in installation and configuration. SAP support may ask users to perform self-checks and send the results.

To ensure that your system is correctly configured for the UI technology of the applications that you want to test, you can display the front-end recorder type and determine the UI technology (SAP GUI, CRM Web Client, Web Dynpro ABAP, for example) of the self-check.

## Procedure

1. In the Administration - Test Suite view, choose CBTA  Perform Self-Checks.

Depending on the SAP Logon configuration, the system asks you to accept requests from the SAP Solution Manager system to your CBTA client. Confirm the message.

You go to the Test Management Self-Checks screen. The system connects to the CBTA client, to determine the self-checks available. This retrieval is dynamic and supports future enhancements and the addition of self-checks via service packs and patches.

1. Choose the Front End Component Self-Checks tab.

If there are no configuration issue, the first self-check, CBTA COM Object Registration, is green, and the version of the client (COM component) is displayed in the Help column.

1. If no self-check is available, either the CBTA client is not installed or there is a configuration issue. Do the following:
   * Install the CBTA client.
   * Repeat the configuration of CBTA, as described under Prerequisites, above.
2. Perform the self-checks.
   * Select the tab page.
   * To retrieve the SUT self-checks, on the System Data Container Checks tab page, select the target component, the logical component, and the system role.
   * Choose Get Self-Checks.
   * Select the available self-checks.
   * Choose Start Execution.
   * To resolve issues, follow the instructions in the Help column.
   * Repeat steps 1 to 4 until all self-checks are performed.
   * To export, on request of SAP support, the result of the self-checks, choose Export All.
3. To identify the UI technology of a self-check, on the Front End Component Self-Checks and System Data Container Checks tab page, see the Provider column.

# Defining the Report Format of the CBTA Execution

## Prerequisites

* To use the MS Word format, you need to at least have Microsoft Office 2010.
* To use the MS Word format with Execution Statistical Graphical, you need to at least have Microsoft Office and Excel 2010.
* To use the PDF format, you have installed at least Microsoft Office 2010with PDF add-on.
* You have executed a test configuration or a test script

Context

Component-based test automation (CBTA) supports three execution report formats:

* The HTML report (default) is displayed on the front end in a browser window.
* The Microsoft Word with Execution Statistical Graphical format report provides additional performance information, for example, a graphic showing the execution time of each test component.
* The PDF format that contains the same information as the MS Word report.

## Procedure

1. In the Administration - Test Suite application, choose CBTA   Maintain Client Settings. Under Microsoft Windows, choose Start  CBTA  CBTA Configuration
2. On the Execution tab, select the report format.
3. Save your entries.

# Client Product Documentation

Once CBTA client is installed, the documentation is available on file system in <CBTA\_INSTALLATION\_FOLDER>\Documentation.

# Recording a Test Script Using Component-Based Test Automation

## Use

You can record business scenarios and create automated test cases with component-based test automation.

You can create tests and components from the screens of a transaction, and parameterize them. These tests are for a single transaction and can be combined into a scenario test. You maintain the components in the test composition environment.

Note

You can test the following types of applications:

* SAP ABAP servers:
  + SAP GUI
  + SAP CRM Web Client
  + Web Dynpro ABAP
  + Business Server Pages (BSP)
  + SAP GUI for HTML
* SAP Java servers:
  + SAP Portal
  + Java Web Dynpro
* Web servers:
  + URL (HTML pages)
  + SAP UI5/FIORI

Note

There are limitations to the scope of CBTA.

## Prerequisites

* You have installed the CBTA client on your local machine
* You have created a test profile.
* You have created a test configuration or a test script and are editing it in the test composition environment.
* You have maintained the attributes of the test configuration or a test script.

## Record Html into SAP GUI Transaction

With CBTA 3.0.8, it is possible to record Html steps embedded into a SAP GUI transaction.

For more details, please refer to [Client Product Documentation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/40ea9a4b83eb4c7ea61a4788f48eb194.html).

Procedure

1. Edit the test script.
2. Choose Launch CBTA.
3. The Test Creation Wizard starts.
   * Optional: Replace the default analysis name with a more meaningful one.
4. To confirm, choose Next.

The executable starts. The Test Creation Wizard is in recording mode.

1. Perform the business process.
2. Optional: To add a checkpoint, do the following:
   * In the Test Creation Wizard, choose Add Checkpoint.
   * Click the UI control to be checked.

The UI control is listed in the test creation wizard.

* + Optional: Change the property to be checked.

When changing this property, its value is displayed.

* + Optional: Change the operator and value to be checked.

1. Optional: To add a screenshot to the execution report, choose Insert Screenshot.
2. To stop the recording, in the Test Creation Wizard choose Stop the PFA and validate.
3. Choose Next.

The structure of the process flow is displayed.

1. Verify that the business process has been recorded correctly and completely .
2. To create the test, choose Next, or

to cancel the recording, choose Cancel.

The test script and test components required are generated and uploaded to the SAP Solution Manager.

1. Choose Finish.
2. In the test composition environment, choose Refresh.

The test composition environment is updated with the results of the recording. The test script parameters, if defined, are displayed on theParameters tab page.

Creating Checkpoints

1. To define a checkpoint, choose Add Checkpoint in the test creation wizard for recording.
2. In the tested application, select the UI element to be checked.
3. In the test creation wizard, a new line has been added to the list of steps. The list of properties that can be checked is automatically retrieved according to the UI element type.
4. If the UI element is still visible in the tested application, the value of the property selected in the wizard will be automatically filled and updated.
5. Select the checkpoint type:
   * Check Data: If you want to perform a check, select Check Data. Under Parameters, select the operator of the check to perform. You can set a value for the check or keep the default, which is retrieved only if the UI control that is targeted by the check is still available in the tested application. If the tested application is updated asynchronously, select Data Updated Asynchronously under Options, to perform the check repeatedly until it succeeds or until a timeout is reached. Use this option when the test execution must wait for an unknown time, for example until a background job has finished. When this option is selected, you can set some additional parameters under Options:
     + Initial Wait Time: The time to wait before the first execution of the check.
     + Interval: The time to wait between two executions of the check.
     + Timeout: The total time during which checks will be performed.

The Options field under Parameters is automatically updated according to the value of these parameters. This value is set for the OPTIONS parameter of the CBTA\_WEB\_CHECKPROPERTY default component instance, which will be inserted in the generated test. You can also define this value later in the test composition environment.

* + Get Data: If you only need this checkpoint to make the UI element property value available for another step, set the checkpoint type to Get Data. No check is performed. You do not need to define an operator and a value, so these fields are disabled under Parameters. ACBTA\_WEB\_GETPROPERTY (or CBTA\_WEB\_GETATTRIBUTE) default component is inserted in the generated test. An output parameter exposes the value of the selected property for the targeted UI element. This value is also stored in the execution context.

Other Functions

* Adding Screenshots: To make the report of the execution of a test more readable, you can insert some instances of the default componentCBTA\_WEB\_CAPTURESCREEN in the test during recording. To append the screenshot step to the recording, choose Add Screenshot. Alternatively, right-click a step under Steps and choose Insert Screenshot. A new step for a screen capture is inserted under Steps.
* Removing Steps: Select a step and choose Remove to delete a step from the current recording. Note that this can result in a non-executable test if you remove a mandatory step.
* Pausing: To pause a recording, choose Pause. To continue, choose Resume.

## Quick Repair

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

You may start the recorder in different modes instead. These modes allow you to maintain the existing test scripts by re-recording only a subset of the scenario. The newly created step can be added at the end of the existing test script or inserted after a particular step.

Recording Options

* Record a new test script and overwrite all existing steps
* Record additional steps and insert them after the selected one
* Record additional steps and replace everything after the selected one

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

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

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

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

# Creating a Login Schema

## Prerequisites

You are editing a CBTA-based test configuration in the test composition environment (TCE).

Note

Only use non-production users and systems.

Do not use a login schema in a test script if its test profile refers to an SUT for which you have configured anonymous URI-based SUT access.

## Context

To be able to automate Web scenarios, the recorder and the execution engine must be able to log automatically into the system under test (SUT), using its login page. Because there are different login pages in the SUTs, the user must define a login schema by specifying Uniform Resource Identifiers (URIs) of the UI elements.

A login schema can define URIs of input fields, buttons, checkboxes, and links, for example.

To make its definition easier, you can use an assistant to define the URI of the login page elements.

With CBTA the NW\_STANDARD\_LOGON login schema is provided, which you can use to record and to execute test scripts on SAP NetWeaver Java systems with at least version 7.01.

All login schemas are stored in the MIME repository (SE80) of the SAP Solution Manager system.

A login schema contains three parts: pre-login steps, login steps, and post-login steps. Each part contains some actions. An action is defined by an URI and an optional value. This value is used when the action needs a parameter, for example to set a value in an input field or to select a checkbox (in that case, the value would be checked). You cannot define values for the user field, the password field, and the login button.

* Pre-login steps

Perform mandatory actions before entering the user name and the password. You can define as many actions as required.

* Login steps

Main part of a login schema. Define three mandatory URIs:

* + Entry field for the user name
  + Entry field for the password
  + Pushbutton or link to activate the validation of the credentials

If the login screen contains other fields (to specify the language, for example), you can define additional URIs. Such actions are performed before the user credentials are confirmed.

* Post-login steps

If some steps are to be performed after the login button has been pressed, you can define post-login actions. You can define as many actions as required.

## Procedure

1. Open the logon page of the application to be tested.
2. Open the test script in the TCE.
3. On the Attributes tab page, under Additional Test Options, choose F4 help the login Sschema .

A dialog box appears.Existing login schemas found in the MIME repository are displayed.

By default, CBTA provides the NW\_STANDARD\_LOGON login schema, which you can use to record and execute test scripts on Java systems (version 7.01 or higher).

1. Choose Create.

The Login Schema Creation Wizard is displayed.

1. Enter data as required.

If you enter the name of an existing login schema, a counter suffix is added by default. To overwrite an existing login schema, choose theOverwrite existing login schema field.

1. In the Login Schema Wizard, choose the Refresh button.
2. In the list of available sessions, select the session corresponding to the logon page of the tested application.
3. Define the URIs of the logon page:
   1. Choose the Spy button of the field that you want to identify, for example the user name field.
   2. Switch to the logon page of the tested application.
   3. On the logon page, position the mouse on an object (in our example, the user name field).

UI elements under the mouse are be highlighted. If UI elements are not highlighted correctly, check the zoom factor in MS Internet Explorer which must be 100%.

* 1. Click the object you want to identify (the user name field, in our example).
  2. Go back to the Login Schema Creation Wizard.

The URI of the selected UI element is displayed in the corresponding field.

* 1. Repeat the previous step for all UI elements to be specified. The following fields are mandatory:
     + Input field for the user name
     + Input field for the password
     + Confirmation button

To specify additional fields (a field to set the language, for example), define additional login steps. To specify the value, choose the With value field.

Note

For the user name field and the password field, the With value fields are disabled since the user name and password are automatically retrieved from the SUT management.

1. Define pre-login and post-logon steps, if required.

If some actions have to be performed before choosing the login button, define additional login steps.

If some actions have to be performed before entering the user name, repeat this operation with a pre-login step.

If some additional actions have to be performed after choosing the login button, repeat the operation with apost-login step.

1. Give a name to the new login schema and an optional description. Note that there is an option to overwrite any existing login schema with the same name. If this option is not selected, a counter suffix will be appended to this new login schema name.
2. Choose Next.

The login schema is uploaded to the MIME repository of the SAP Solution Manager system.

The Object Directory Entry Creation dialog box appears.

If the dialog box is not visible in the Windows task bar or if the wizard seems to be blocked on the upload step, minimize all others applications, to display the dialog box.

1. Select a transport package, and save your entries.
2. Go back to the Login Schema Creation Wizard window.

The status is green.

1. Choose Finish.

## Results

The login schema can be selected in the corresponding field when defining the test script attributes in the TCE:

At recording and execution time, the login schema file is downloaded to the front-end environment, next to the CBASE files. It is then used at login time.

Login schema definition files (with the .lsd extension) are stored centrally in the SAP Solution Manager, and synchronized automatically on all front end machines that use CBTA. It is stored in the mime repository, next to the CBASE files. If your user has the S\_DEVELOP role, you can display them in transaction SE80, under SAP/PUBLIC/CBTA/LOGIN\_SCHEMA.

# Editing a CBTA Test Script

## Context

When you have recorded a test script, you can edit and complete it.

You can edit components of a test script that was recorded with the Component-Based Test Automation (CBTA) tool in the Test Repository views.

A CBTA test script consists of two types of components:

* Default components are delivered by SAP with the CBTA software. They are stored in the SAP Solution Manager repository for eCATT test scripts. They perform single standard actions, for example, entering a value in a field, pressing a button, or selecting a tab page.
* Screen components are generated after the recording of a test, depending on the screens that were used in the recorded process step. The inspection process retrieves static information on the screen from the system under test. It generates a screen component for each screen, which is a logical part of a page. A screen component contains one parameter for each field of the screen. When a value is assigned to such a parameter, the system enters it in the corresponding field, at execution time.

A screen component reflects a single version of a screen. Different versions of a screen, for different tests, are stored in different logical components for different logical component groups. So test management can differentiate between different versions of the same screen. The version of a screen is the same on all systems of the same logical component. Otherwise, screen components are created.

* CBTA script, like a component, allows to reference another CBTA test script content and during execution to perform the child CBTA script into main CBTA test script. From end user point of view, this feature allows to create some group of default component and to reuse into different CBTA scripts. Example: CBTA script is created to close a popup if the popup is visible before. To start the scenario, this script can be included in all other CBTA scripts like a first component of test scripts.

You can reuse default and screen components in several test scripts. When a screen is updated, its screen components can be repaired, to make the repair of test scripts easier. This process uses the technical bill of material (TBOM) and business process change analyzer (BPCA), to reduce the effort of testing after a system update or upgrade.

## Procedure

1. In the Test Repository - Test Configuration view, select the test configuration that includes the test script you want to edit.
2. Choose the Test Script tab.

The test script steps are displayed. In a CBTA test script, steps are either default or screen components.

You have the following options:

* + In the Comment column, you can enter or edit a description of the test script step.
  + In the State column, you can disable a test script step temporarily. During test executions, the step is skipped, so you can, for example, try various variants of a test script without having to remove steps.

1. You can add test script steps, under Test Script Steps:
   * 1. In the Type column, select a component type:
        + Use Default Component to perform basic actions, such as entering a value in a field, selecting a tab page, pressing a button, or clicking on a link.
        + Use Screen Component to set the values of a set of fields in one screen (SAP GUI) or views (CRM Web Client or Web Dynpro ABAP).

To identify the screen or views of a field, do the following in the CBTA tool, choose the Get Additional Information button. Then, select the session showing the field and select the field with your mouse pointer. The technical information is displayed: the URI that identifies the field, and the screen (program and screen number for SAP GUI) or the application and view (CRM Web Client and Web Dynpro ABAP).

To use this information to create a screen component, choose CBTA Inspection.

* + 1. In the Name column, select the test script.

Save your entries. Empty lines are removed automatically.

In the Use CBTA field, enter the name of the test script referenced.

1. To define the sequence of test script steps, use the Up and Down buttons.

To duplicate several steps with a test script, you can use the copy and paste feature.

To remove a superfluous step (for example, because you have performed it twice, by mistake) choose Remove.

1. Configure parameters under Test Script   Parameters:
   * To define the values passed to the test script steps (import and export parameters), edit the Parameters sub tab page:
     1. In the Usage field, define the usage type:
        + No value: No parameter value is passed to the parameter of the test script. The test script parameter does not have a default value.
        + Fixed: A constant value is passed to the parameter. The constant value does not depend on the execution of the script.
        + Local: The value passed to the parameter is mapped into the parameter of another step. The value actually used during the execution of the test script depends on the execution of the parent script/previous step and the values created in the preceding steps.
        + Exposed: The parameter is mapped to the parameter of the parent script. The parameter value is passed from (import parameters) or passed to (export parameters) the parameter of the parent script.

Import parameters can receive the values from the test data container.

Several exposed import parameters can receive their values from the same parent script parameter.

A single exposed export parameter can pass its value to several export parameters of the parent script.

## Example:

**Fixed scope for parameters**

You always want to create a sales order for 5 pumps, enter 5 for the parameter Quantity. The values will still be fixed for each test script execution.

In the Value column, set usage to Fixed and enter your fixed value (ex: 5). To define an empty string value, use the value %BLANK%.

Local scope for parameters

You want to create a main test script comprising 3 steps (type: BCTA test scripts) with the following parameters:

* Create quotation
  + Import parameters: Customer, Material, Quantity
  + Export parameter: Quotation Number
* Create sales order from quotation
  + Import parameter: Quotation Number
  + Export parameter: Sales Order Number
* Create delivery from sales order
  + Import parameter: Sales Order Number
  + Export parameter: Delivery Number

Put 3 of the steps in the sequence. To set the usage local on parameters, you can link export parameters from Step Create quotation – Quotation number to Step Create sales order from quotation - Quotation Number, … during the test execution when the first test script will be finished the quotation number retrieved will be passed to second test script into the quoatation parameter.

# Working with Message Parameters

## Use

To retrieve data from messages for the test automation of Web and Web Dynpro scenarios, you can use message parameters.

You can retrieve information from a message that the application displays in the message area, with the default componentCBTA\_WEB\_A\_GETMESSAGEPARAMS. This component is not used by default when you record the tested business scenario. As the test engineer, you must manually add it to the generated test script. It allows searching the message area for a message. The message sentence must match the pattern that is specified as input parameter. The sentence is parsed and its fragments are exposed as output parameters.

Example

The Create Shopping Cart scenario typically ends with a message that informs the user about the creation status. This message contains the number of the new shopping cart. You retrieve it and pass it to the subsequent steps of the test script.

## Prerequisites

You have installed and configured CBTA 3.0 SP02 or higher.

You have SAP Solution Manager 7.1 SP10 or higher.

## Procedure

Default Component CBTA\_WEB\_A\_GETMESSAGEPARAMS

* Input Parameters
  + URI (optional): the URI identifying the message area

If empty, the component searches for the message in the main document of the main window of MS Internet Explorer.

* + MessagePattern: the pattern that the message must match
  + Options (optional): /u (uppercase) to ignore the case when using the pattern
* Output Parameters
  + MessageText: provides the full message text
  + MessageParameter1: provides the fragment matching placeholder {1}
  + MessageParameter2: provides the fragment matching placeholder {2}
  + MessageParameter3: provides the fragment matching placeholder {3}
  + MessageParameter4: provides the fragment matching placeholder {4}
* Placeholders

The syntax of the message pattern can include placeholders to extract some words from the text that is displayed in the application:

* + {?}: matching one or more words. Use this placeholder to ignore part of the text.
  + {1}: matching one or more words and exposing them by MessageParameter1
  + {2}: matching one or more words and exposing them by MessageParameter2
  + {3}: matching one or more words and exposing them by MessageParameter3
  + {4}: matching one or more words and exposing them by MessageParameter4

Example

To retrieve a shopping cart number you can use, for example, the following message pattern:

Shopping cart "{?}" with number {1} saved successfully

With this pattern, the words providing information about the user and the date are ignored, and the shopping cart number is exposed by the output parameter MessageParameter1.

Alternatively, you can retrieve the information with the following syntax:

Shopping cart "{1} {2} {3}" with number {4} saved successfully

In this case, we get the following output parameter values:

MessageParameter1: TESTER\_01

MessageParameter2: 20.09.2013

MessageParameter3: 22:02

MessageParameter4: 1000133807

Configuring the Test Script

1. Configure the test script.

In the Create Shopping Cart scenario, the application is embedded in a portal page.

The recorded scenario ends with the message that contains the shopping card number. By default, there is no step to retrieve the shopping card number that we need, to check whether the test execution fails. If you have defined a checkpoint during recording (using the Check Picker), it fails because message text contains information that always changes, for example the date and the time.

1. To retrieve the URI of the message area, use the object spy.

Note

CBTA uses the URI to identify UI elements in the HTML content. For most of the components, the value of the URI parameter is automatically populated by the test recorder and does not need to be adapted. The message area is an exception. The messages that are displayed here do not have stable IDs and they cannot be found using the regular approach.

* 1. To start the object spy, in the Test Composition Environment, choose Get Technical Information.
  2. Specify the message area.

HTML attributes and properties are displayed in a tree. In our example, the Create Shopping Cart application is a WebDynpro (ABAP) application based on the Unified Rendering (Light Speed) framework. The URI of the message area is provided in the Light Speedcontext node of the tree.

Adapt the test script:

* 1. If you have defined a checkpoint during recording, disable the CBTA\_WEB\_CHECKPROPERTY component.
  2. To invoke the CBTA\_WEB\_A\_GETMESSAGEPARAMS component, add a new step.
  3. Specify the URI of the message area that you have determined, with the object spy.
  4. Use placeholders to extract the required fragments from the text of the message.

1. To check whether the message data is retrieved correctly, execute the test script.

The CBTA\_WEB\_A\_GETMESSAGEPARAMS default component retrieves the information from the message and makes it available to subsequent steps by its output parameters. The information is displayed in the execution report.

For more information about using output parameters, see the CBTA How-to Guide at [http://wiki.scn.sap.com/wiki/x/kQ\_IFQ[Information published on SAP site](http://help.sap.com/disclaimer?site=http://wiki.scn.sap.com/wiki/x/kQ_IFQ)](http://help.sap.com/disclaimer?site=http://wiki.scn.sap.com/wiki/x/kQ_IFQ).

The same information is also accessible using the concept of tokens, as shown in the last graphic. Tokens pass 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.

# Creating Screen Components for ABAP Web Dynpro Applications

## Prerequisites

You have created a test script with an ABAP Web Dynpro executable object.

The ABAP Web Dynpro executable object includes the name of the application.

## Context

When you need to manually edit a test, it can be necessary to generate screen components manually. This document describes this activity for a Web Dynpro ABAP application.

A screen component can be inserted as one step of a test script. Each parameter of a screen component corresponds to an input field, such as dropdown menu, check box, or radio button, in the automated user interface. At execution time, each parameter that has been provided with a value is populated with this value in the corresponding user interface.

You can reuse screen components in several tests. This makes it easier to repair tests if systems are updated with new service packs or custom developments.

Note

You can only reuse screen components in tests that use the same logical component. This ensures that tested products have the same version. If no logical component is used in the target component of the system data container of the test, the product versions of the system under test are used to assemble compatible screen components.

Creating screen components manually is an advanced function. We recommend that you automatically record test scripts with CBTA recording wizard.

For ABAP Web Dynpro applications, the screen components are generated based on statically defined Web Dynpro views, from the same or a used component.

If a Web Dynpro view is generated dynamically at runtime, you cannot create a screen component. Use a default component instead.

## Procedure

1. To identify the name of the relevant view, in the Web Dynpro application UI, choose Technical Help in the context menu of a field, and take note of the following:
   * Web Dynpro component displayed under Web Dynpro Application  Start Component
   * View name displayed under Views and View Elements
2. In the Test Repository applications, open the test script or test configuration for editing.

To inspect the relevant view, choose CBTA Inspection on the Test Script tab. The Inspection Wizard is started.

1. Select the view and choose Next.

The screen component is generated

1. Choose Finish.
2. Add the screen component, and the corresponding parameters, to the script.
3. Identify the parameter value of the parent context URI (PARENTCONTEXTURI).

Caution

* + The PARENTCONTEXTURI parameter is specific to ABAP Web Dynpros. It is necessary for the usability of the screen component.
  + Use the Object Spy function to retrieve the identifier (URI) of a user interface control. Use it only in applications with stable IDs (parameter sap-wd-stableids = X must be set in the URL).
  + Open the application that you want to test.
  + Go back to the editor for test scripts. On the Test Script tab, under Parameters, choose Get Technical UI Information.

Go to the Object Spy.

* + On the Web Application tab page, select the session that you previously opened.
  + Choose Spy.
  + Position the cursor on a field in the Web Dynpro view. The field is highlighted in red.
  + To display the value of the PARENTCONTEXTURI parameter, in the Property Name tree go to Light Speed UI  Light Speed Context  Parent Context Uri.

1. Enter the value of the PARENTCONTEXTURI parameter in the parameters of the screen component.

# Running a CBTA Test Script in Debug Mode

## Use

If a component-based test configuration (CBTA) test script fails, you can execute the components (test steps) individually, one by one. CBTA provides a graphical overview of the test steps and displays the corresponding UI components attributes.

With the CBTA debugger you can debug a test at component level, without seeing the Visual Basic Script (VBS) code, so you see the actual state of the tested application UI, as well as the values of the input and output parameter.

## Procedure

1. Start the Test Repository - Test Scripts application.
2. Select a CBTA test script, and choose Execute.

You go to the Start Options screen.

1. On the UI Control tab , in the Mode for External Tool field, select Debug Mode and choose Execute.

The CBTA debugger starts. A graphical overview of the test steps is displayed.

1. You have the following options:
   * To insert breakpoints, select one component and choose the Toggle Breakpoint button. A breakpoint is represented by a red dot next to the component name.
   * To change parameter values dynamically, double-click them in the table, edit the new value and choose Enter.
2. To execute the test script steps while the corresponding UI components are displayed, choose Run or Step Over.
   * If you choose Run, all the components are executed until a breakpoint is reached, an error occurs, or the last step has been performed.
   * If you choose Step Over, only one component is executed. The debugger stops the execution on the next component.

The test script steps are executed while the corresponding UI components are displayed.

1. You have the following options:
   * To insert breakpoints, choose Toggle Breakpoint. The following shortcuts are available:
     + Toggle Breakpoints: F4
     + Run: F5, F8
     + Skip a component: F6, F10
   * Input and output parameters are displayed in the corresponding tabs on the right-hand side. You can change input parameters of the current component by double-clicking a value, and editing it. The changes are taken into account for the current component execution. The tab for internal parameters shows all parameters and variables that are currently defined in the CBTA runtime.
   * To identify errors in the CBTA log (on request by the SAP support, for example), choose Logs. Alternatively, in MS Windows, choose Start  All Programs  CBTA  CBTA Configuration. On the Log tab, choose View Log.
   * Tokens are special variables that can be used by tests using the %Token\_name% syntax. The corresponding tab is read-only.
   * Conditions show the assertions that are currently met. This is used by the IF default component or statement and by theSetCondition default component.
   * The Report tab dynamically shows the HTML type report that is constructed after each component execution.
   * When complex errors occur, it can be required that SAP support asks you to provide client-side log files. These files are under Logs in the CBTA debugger. They can also be found in the front-end configuration windows.

Error Behavior

In case of an error during the execution, a message is displayed in the status message at the bottom of the CBTA debugger window. You can see more information in the tooltip of the status bar message (hover your mouse over the error message).

When an error occurs, choose Run to end the execution and go back to the SAP GUI session with the Start Options screen to display the execution report.

Using the Object Spy

Some test executions may not work due to a wrong usage of the Unified Resource Identifier (URI). A URI is a string that is used to uniquely identify a control of the automated page. With the object spy, you can retrieve the URI of a control of a page:

1. Choose Object Spy, select the tab corresponding to your UI technology (SAP GUI for the first tab or Web application in the second tab).
2. Refresh the list of existing applications, and select the right one.
3. Then choose Spy and select with your mouse cursor the control of the page that you are interested in. Click this control.
4. Corresponding information is displayed at the bottom of Object Spy window.

Note

The spied session is not opened by the object spy. Reuse an existing one or open a new session. To use the object spy on CRM Web UIs, certain parameters need to be added to the URL, for example sap-testmode=T&sap-titleprefix=%5bTAF%5d.

For Web Dynpro ABAP, enable the stable ID mode beforehand. Use either the URL parameter sap-wd-stableids=X or use the user parameter WD\_STABLE\_ID\_F\_TEST = X (transaction SE3). For both CRM Web UI and Web Dynpro ABAP, those parameters must be set before the user is logged in.

# Displaying the Execution Report

## Use

Test execution reports provide detailed information about the test execution, performed steps, parameters, and errors. They tell you what a test did and why it has failed.

## Prerequisites

You have executed a test configuration or a test script.

## Procedure

In the Test Repository applications, select a test script or test configuration for editing.

In the editor, choose Display Log:

* Choose Test Tool Log for tests made for the component-based test configuration (CBTA) or for third-party test tools.

By default, the CBTA execution report is in HTML format, but you can choose the Microsoft Word format. Choose Solution Manager Log for tests made for eCATT or composite tests. With this option you can also access to the log for other tests.

# Customization

# Using Runtime Library Manager

# Enhancing CBTA With Custom Code

Certain actions cannot be performed with the default component-based test automation (CBTA), for example recording or playing a test with Adobe Flash technology, with Java applets, or with Web controls that have complex javascript handling. You can enhance CBTA with custom functions. The CBTA engine is based on a Visual Basic Scripting (VBS) library called 'runtime library' (RTL). You can add custom code to this library using the CBTA runtime library manager.

## Terminology

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

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

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

Query API: When you write custom functions for Web applications, as the test engineer, you may need access to the HTML content that the application displays in the Microsoft Internet Explorer. To create a new custom function to address test automation issues, download the runtime library from the SAP Solution Manager system, and make it available on the local file system of your machine. Note: When the RTL manager starts for the first time, no customization is active, so the execution of a CBTA test script has previously relied on the official runtime library that SAP delivers.

## Prerequisites

* You have enabled the RTL manager web service.
* You have the authorization for the RTL manager.
* You have read SAP Note.

# Starting the Runtime Library Manager

## Context

## Procedure

1. In the Administration - Test Suite view, choose CBTA  Customize RTL.In the RTL manager logon screen, enter your credentials. You are connected to the SAP Solution Manager system in which the CBTA runtime library is stored.
2. Choose Open for Edit
3. Specify where to store the RTL on the local file system. Choose Ok to proceed.
4. An additional panel appears in the RTL manager. It displays the state of the customization (Pending) and the selected location.
5. To navigate to the folder in which the custom code is to be stored, choose Explore.
   * You find the following data:
     + The root folder CBTA\_Customization contains a ZIP file with the official RTL (**CBASE.zip**).
     + The CBASE folder contains the CBASE configuration and the VB script files (\*.vbs). The content of the**CBASE.zip** file is extracted automatically.Note

Do not modify the CBASE configuration and the VB script files. Only VB script files stored in the Custom folder are to be modified.

* + - The sub-folder Custom contains the custom code that has been downloaded from the SAP Solution Manager system. The first time, this folder can be is empty (or can even be missing). It is created automatically when you choose Explore.

# Creating Custom Functions Manually

## Context

## Procedure

1. Create the custom VB script file or modify an existing one. You can use any text editor to write the code.

Ensure the following:

* + The VB script files have the extension .vbs.
  + The library has a meaningful name to avoid conflicts with other libraries.
  + The code does not declare global variables.
  + Function and variable names start with a customer-specific prefix.

1. Store your custom VB script files in the Custom folder (for example under C:\XBTA\_Customization \CBASE\Custom). The custom VB script files are displayed on the Functions tab page. If the Custom folder is empty, the Functions tab shows an empty list.
2. To check the syntax, start the VB script file manually, by double-clicking the file name.
3. Choose Refresh

# Using Custom Code Patterns in CBASE Customization

CBTA provides you with a set of patterns that the test engineer can reuse and adapt to create custom functions. In the Runtime Library Manager, you can write additional custom functions to be used in the test scripts to automate a test of business scenarios for which the common approach (based on default components) is not sufficient.

* SAP GUI: UI framework used by SAP applications
* SAP UI5/FIORI
* WebCUIF: Layer which generates the content displayed by SAP CRM applications.
* Unified Rendering Light Speed (LS): Layer which generates the content displayed by SAP applications, such as:
  + Web Dynpro application (ABAP)
  + Web Dynpro applications (Java)
  + SAP GUI HTML – SAP GUI content displayed in MS Internet Explorer
* BSP: applications based on Business Server Pages
* Java Web Dynpro: layer previously used by Java Web Dynpro applications
* Plain HTML pages: content generated using regular HTML tags (with no, or few, scripting capabilities)

For test automation, each UI technology implicates specific challenges, most of which the test recorder and test player can handle. However, if the business scenario is highly dynamic, the default behavior is not always sufficient, so the test engineer needs to write custom functions.

## Prerequisites

* You have configured CBTA and installed the CBTA test tool on the client.
* You have read SAP Note [1778899 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1778899)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1778899) (CBTA Collective Note).

## Process

The following patterns are delivered by default with CBTA. SAP can deliver additional patterns via SAP Notes, depending on customer’s needs.

# Pattern for Web Dynpro ABAP: CheckForMessages

This function searches for application messages in a message container. It verifies whether the applications report feedback, and checks for errors and warnings, for example.Note

The CBTA\_WEB\_A\_GETMESSAGEPARAM default component retrieves message parameters. It is generic and supports all Web UI technologies, but it cannot retrieve WebDynpro-specific information, such as the message type (such as error, warning, info).

CBTA cannot determine whether these messages are normal in the current business context, so the test script at runtime ignores them and continues. These messages may be unexpected, and there are situations in which you should verify whether the test script can continue. As the test engineer, you can create the custom function to check the content of the message area, from the pattern, in the runtime library manager.

Signature of CheckForMessages Pattern

* Input parameters
  + MessageContainerUri

URI of the HTML container displaying application messages. If empty, the function searches for messages in the message area displayed by the main document of the main window.

Default value:: ls.rid=WDR\_MESSAGE\_AREA.MESSAGE\_AREA.ROOTUIELEMENTCONTAINER

* + MessageType

The message type parameter can have the following values: E = error, W = warning, and I= information

* + MessageRule

The message rule parameter can have the following values: F = forbidden, R = required, and A = allowed

* + MessagePattern

Not used – reserved for future use.

* + Options

The option parameter can have the following values: /C = capture a screenshot and /e = capture a screenshot of the message area only.

* Output parameters
  + Output = number of messages matching the criteria

Use the CheckForMessages pattern to check whether application errors are being displayed, and to report any test execution failure. For example, the input parameters could be as follows:

* MessageContainerURI = ls.rid=WDR\_MESSAGE\_AREA.MESSAGE\_AREA.ROOTUIELEMENTCONTAINER
* MessageType = E
* MessageRule = F
* MessagePattern = [empty]
* Options = /c /e

Creating and Using a Custom Function

Assume that the custom function has been created by the runtime library manager, and the corresponding library Custom\CheckForMessages.vbs has been generated. The runtime library manager displays the following:

...

A CBTA test script can use the custom function, but it is not visible by default. You can call it by using one of the following components:CBTA\_WEB\_A\_INVOKE\_FUNCTION and CBTA\_WEB\_A\_EXECUTE\_STATEMENT.

In the following example, the CBTA\_WEB\_A\_INVOKE\_FUNCTION calls the custom code and checks for application errors:

...

To use the CBTA\_WEB\_A\_INVOKE\_FUNCTION component, the test engineer must specify the name of the custom library, the name of the custom function, and the values of its input parameters.

...

In this example, the following parameters are used:

* LIBRARY = Custom\CheckForMessages.vbs
* FunctionName = CheckForMessages
* PARAMETER1 = empty because the MessageContainerUri parameter is optional
* PARAMETER2 = message type E for Errors
* PARAMETER3 = message rule F for Forbidden
* PARAMETER4 = N/A
* OPTIONS = /c /e to capture a screenshot of the message area

When you set MessageType to E (error) and MessageRule to F (forbidden), the test will fail as soon as the message container holds one or more applications errors.

The implementation has two functions (both names are replaced by the function name that the test engineer specifies when he creates the custom function):

* The PATTERN\_NAME function is a façade of the implementation; it includes exception handling which provides human-readable feedback in case of scripting errors.
* The PATTERN\_NAME\_Impl function is the implementation.

Syntax

Code Syntax

Function PATTERN\_NAME ( MessageContainerUri, MessageType, MessageRule,

MessagePattern, Options)

On Error Resume Next

EventComponentBegin()

If ConditionsManager().CheckConditions() Then

PATTERN\_NAME = PATTERN\_NAME\_Impl ( MessageContainerUri, MessageType, MessageRule, MessagePattern, Options )

End If

EventComponentEnd()

End Function

Function PATTERN\_NAME\_Impl ( MessageContainerUri, MessageType, MessageRule,

MessagePattern, Options )

...

End Function

All SAP-delivered default components are built using this approach. This structure is recommended for all custom code. It includes the exception handling, and calls two functions that are required to make the custom code compliant with CBTA requirements.

## Initialization Phase

The first statements (ImportLibrary...) of the implementation are only used to import the library the implementation depends on.

The custom function (PATTERN\_NAME\_Impl = "") can return a value, which is made available to the subsequent steps of the test scripts via the output parameter of the component. The code initializes the returned value to an empty string:

The next statements (IF ... END IF) validate the input parameters, and set them to their default values if they are not defined by the test script.

The function writes information to the technical traces (associated with the execution report), using the ReportDebugLog method. This can help you to troubleshoot execution problems.

Code Syntax

ImportLibrary "WEB\_WebController.vbs"

ImportLibrary "WEB\_WebControls.vbs"

CBTA\_WEB\_A\_INVOKE\_FUNCTION

PATTERN\_NAME\_Impl = ""

If IsNull(MessageContainerUri) Then

MessageContainerUri="ls.rid=WDR\_MESSAGE\_AREA.MESSAGE\_AREA.ROOTUIELEMENTCONTAINER"

Code Syntax

End If

If IsNull(MessageType) Then MessageType = ""

End If

If IsNull(MessageRule) Then

MessageRule = "F"

End If

## Implementation

The code shown below relies on the Query API to access the MS Internet Explorer window and search for UI elements in the HTML content.

For information on the Query API, see

Code Syntax

' Preparation of the Query

Dim query, filter Set query = LsController().CreateQuery()

query.ParentControlUri = MessageContainerUri ' Specifying

the criteria (using filters and conditions) Set filter =

query.SetFilter() filter.AddCondition "tag", "=", "IMG"

filter.AddCondition "ls.type", "=", "LN" filter.AddCondition

"ls.application", "=", "WDR\_MESSAGE\_AREA"

filter.AddCondition "ls.view", "=", "MESSAGE\_AREA"

filter.AddCondition "ls.field", "=", "MSG\_LIST\_ICON"

The query searches for images (<IMG> HTML Elements) that are children of the message container. The criteria also define conditions to only select the images that represent the message types. This is why Light Speed attributes are used here:

* ls.type: type of the Light Speed control; LN for link
* ls.application: Web Dynpro application name
* ls.view: name of the view used to display application messages
* ls.field: name of the field in the view

Additional conditions are set, depending on the message type that you are looking for. The conditions below check the name of a GIF file that represents the type of the message.

Rely on this information to determine the type of the message because this is the only information that the Light Speed framework provides. So this code has not been delivered as a default component.

Code Syntax

If MessageType="E" Then

filter.AddCondition "src", "{endsWith}", "ErrorMessage.gif" ElseIf MessageType="W" Then

filter.AddCondition "src", "{endsWith}", "WarningMessage.gif" ElseIf MessageType="I" Then

filter.AddCondition "src", "{endsWith}", "SuccessMessage.gif" Else

' No filter - This will show all messages (and count them)

End IF

The query is now defined. The next statement resolves the query and retrieves a collection of Light Speed controls that match the criteria:

Code Syntax

Dim controlCollection

Set controlCollection = query.Select()

If controlCollection Is Nothing Then ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - Message container not found" Else ...

End If

If the collection is not empty, the implementation performs the verifications according to the MessageRule specified. The ReportLog method writes human-readable feedback in the execution report.

Code Syntax

If MessageRule="R" Then 'Required

If controlCollection.Count = 0 Then ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - No message of type: " &

MessageType

Else ReportLog "PASSED", "PATTERN\_NAME", "Message Type: " & MessageType & \_ vbCrLf & "Number of Messages : " & controlCollection.Co End If

ElseIf MessageRule="F" Then 'Forbidden

If controlCollection.Count = 0 Then

ReportLog "PASSED", "PATTERN\_NAME", "No messages of type: " & MessageType

Else

ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - Message Type: " & MessageType & \_ vbCrLf & "Number of messages: " & controlCollection.Cou End If

ElseIf MessageRule="A" Then 'Allowed

ReportLog "INFO", "PATTERN\_NAME", "Number of messages: " & controlCollection.Count

End If

This following section iterates through the collection of controls, to retrieve the text of the message associated with the IMG HTML element. TheGetProperty method retrieves the innerText HTML property containing the text being displayed.

Code Syntax

Dim childControl, associatedControl

For i=0 To controlCollection.Count-1

Set childControl = controlCollection.ControlAt(i)

If Not childControl Is Nothing Then Dim imgId,

spanId imgId =

childControl.GetRelevantControl().GetLsId()

Code Syntax

spanId = Replace( imgId, "MSG\_LIST\_ICON", "MSG\_LIST\_TEXTVIEW" )

Set associatedControl = childControl.FindElementById( spanId )

If Not associatedControl Is Nothing Then ReportLog "INFO",

"PATTERN\_NAME", "Message #" & (i+1) & " - " & \_

associatedControl.GetProperty( "innerText" ) End If

End If

Next

The last part checks the Options parameter and captures a screenshot by calling the WEB\_CaptureScreen method (associated to theWEB\_CAPTURESCREEN Default Component).

Finally, the function sets the PATTERN\_NAME\_Impl to return the number of messages matching the MessageType.

Code Syntax

If InStr(Options, "/c") Then

If InStr(Options, "/e") Then WEB\_CaptureScreen MessageContainerUri, "/e"

' Captures the Message Container only Else

WEB\_CaptureScreen MessageContainerUri, "" ' Capture the whole browser window

End If

End If

PATTERN\_NAME\_Impl = "" & controlCollection.Count

1. Pattern for CRM Web Applications: SelectMenuItemByText

Some CRM applications associate a context menu that lets the user perform additional actions, to some controls. The test recorder can normally record such actions and perform them at runtime, by opening the context menu and simulating a mouse click the child item.

Unfortunately, for some scenarios the default behavior does not work, because the ID associated to the child item is not stable: The ID is different each time the CRM application runs.

So we search for the child item by text, using the SelectMenuItemByText pattern.

Use the CRM application, for example, to create an incident: business role SOLMANPRO and logical link SM-IM-CR.

The Object Spy or the Microsoft Developer Tool (F12 in MS Internet Explorer) indicate that the ID includes a server-generated GUID, so you cannot search for the item by its ID.

Signature of SelectMenuItemByText Pattern

Searches for application messages in a messages container. This function verifies whether the applications does report feedback, and checks for errors and warnings, for example.

* Input Parameters
  + Uri: URI of the control with which the context menu is associated
  + Text: text of the child item that is to be selected. This text is case-sensitive except when the /u option is specified.
  + Parameter3: not used
  + Parameter4: not used
  + Options: several options influencing the behavior are available.
  + /u = upper-case, item selection is then case-insensitive
  + /click = When this option is set, the implementation opens the context menu by simulating a mouse click on the main control, which is not done by default because a previous step of the test may have already done so.
* Output Parameters
  + Output: set to DONE when the action has been performed. It supports the following values:

SelectMenuItemByText has 5 parameters to comply with the default component for calling custom functions:

CBTA\_CRM\_A\_INVOKE\_FUNCTION. Leave Parameter3 and Parameter4 empty when calling this custom code.

Similar to all other patterns, the implementation consists of two functions. The first one is a facade to the implementation, which includes execution error handling statements. The second method actually performs the job.

The PATTERN\_NAME fragment (see below) is automatically replaced by the final custom function name when using the Code Assistant feature of the runtime library manager.

Code Syntax

Function PATTERN\_NAME ( Uri, Text, Parameter3, Parameter4, Options )

On Error Resume Next ' Important - Exception Handling - Do not change it!

EventComponentBegin()

If ConditionsManager().CheckConditions() Then PATTERN\_NAME =

PATTERN\_NAME\_Impl ( Uri, Text, Parameter3, Parameter4, Options )

End If

EventComponentEnd()

End Function

Function PATTERN\_NAME\_Impl( Uri, Text, Parameter3, Parameter4, Options ) 'internal

ImportLibrary "CRM\_WebController.vbs"

ImportLibrary "CRM\_WebControls.vbs"

PATTERN\_NAME\_Impl = "NOT DONE"

...

End Function

The first statements of the implementation initialize local variables, validate the input parameters, and write information to the traces, for troubleshooting:

Code Syntax

Dim bodyControl, bodyElement, collection, childElement, anchors, anchor, currentText

If IsNull(Uri) Then

ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - Button Uri not specified"

End If

If IsNull(Options) Then

Options = ""

End If

The next section checks the Options parameters, to determine whether it is necessary to open the context menu:

Code Syntax

If InStr(Options, "/click" ) Then

CRM\_WebControl\_Click Uri

End If

Note

The following implementation invokes the CRM\_WebControl\_Click method, which corresponds to the default component CBTA\_CRM\_CLICK.

Once the menu item is opened, the implementation can continue and search for child items. The following code first accesses the body of the work area, using the GetCrmControlByUri method that the CRM application controller exposes. This step waits for documents to be ready:

Code Syntax

' Gets the body of the current workarea

Set bodyControl = CrmController().GetCrmControlByUri("crm.area=WorkArea; tag=body")

When the body of the document is ready, the content of the menu should be visible and the implementation can proceed. The following statements access the body HTML element using the GetHTMLElement method, and search for SPAN HTML elements using the MSHTML Microsoft API:

Code Syntax

' Gets the body of the current workarea

Set bodyControl = CrmController().GetCrmControlByUri("crm.area=WorkArea; tag=body")

If bodyControl is Nothing Then

ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - Workarea not found"

Else

Set bodyElement = bodyControl.GetWebControl().GetHTMLElement()

If bodyElement is Nothing Then

ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - Workarea not found"

Else

Set collection = bodyElement.all.tags("SPAN")

If collection is Nothing Then

ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - No SPAN HTML Elements"

Else

ReportLog "DONE", "PATTERN\_NAME", "Count : " & collection.length

...

End If

End If

End If

The following statements do not rely on CBTA interfaces. The bodyElement implements some scripting interfaces (from the MSHTML API) and the all and tags properties return a set of objects that also implement scripting interfaces.

Code Syntax

Set collection = bodyElement.all.tags("SPAN")

The collection variable contains a list of SPAN HTML elements in the work area of the CRM application. The implementation filters out the SPANHTML elements that are not menu items by verifying whether the SPAN element has an ID including the T\_MENU\_ITEM fragment, and that it has at least one child element of type ANCHOR (<A> HTML tag).

The visible text of the item is associated with the ANCHOR element. Its value is retrieved by innerText property, and checked against the expected value.

Code Syntax

For i=0 To collection.length-1

Set childElement = collection.item(i)

If Not IsNull(childElement.id) Then

If InStr( childElement.id, ":T\_MENU\_ITEM:") Then

Set anchors = childElement.all.tags("A")

If anchors Is Nothing Then ReportLog "FAILED", "PATTERN\_NAME",

"Operation Failed - No Anchor HTML

Elements"

Else ReportLog "DONE", "PATTERN\_NAME", "Child Anchors -

Count : " & anchors.length Set anchor = anchors.item(0)

CBTA.Report CBTA.INFO, "PATTERN\_NAME", "Anchor Element - id=" & anchor.id,

""

currentText = anchor.innerText

If InStr(Options, "/u" ) Then

currentText = UCase(currentText)

Text = UCase(Text)

End If If InStr(

currentText, Text ) Then

anchor.FireEvent("onmousedown")

anchor.setActive()

anchor.focus()

anchor.FireEvent("onmouseup")

anchor.click()

PATTERN\_NAME\_Impl = "DONE"

Exit For

End If

End If

End If

End If

Next

If the text matches the one searched for, the implementation simulates a mouse click on the item, calling several methods exposed by the MSHTMLAPI:

Code Syntax

anchor.FireEvent("onmousedown")

anchor.setActive()

anchor.focus()

anchor.FireEvent("onmouseup")

anchor.click()

Instead of calling methods exposed by the MSHTML API, the implementation could reuse the CRM\_WebControl\_Click method to select items, for example as follows:

Code Syntax

Dim anchorUri anchorUri = "label=" & anchor.innerText & ";

crm.area=WorkArea; tag=A; id=" & anchor.id

CRM\_WebControl\_Click anchorUri

**Limitations**

The SelectMenuItemByText pattern searches for an HTML element in the body of an HTML document. The current implementation has the following limitations:

* It does not support the selection of subitems (children of another child item).
* It only searches for items in the main work area of the CRM application. The code must be adapted if the item is shown in the HTML document of a modal popup.
* The context menu associated to the main control is opened by mouse click. Some controls may require a right-click.
* The code shown here assumes that the application can only display a single context menu, so it does not verify whether the item is associated with the main control.

1. Pattern for CRM Web Applications: SelectTransactionType

The SelectTransactionType pattern searches for and selects a transaction type in the popup window , which appears when starting some CRM applications.

Some SAP CRM applications start by prompting the end user for a transaction type. This function performs this action by searching for the transaction type by its text, wherever it is in the list.

Note

This pattern only has the purpose of illustrating how to solve this problem and to explain to the Query API that

CBTA can solve such issues. The equivalent component is CBTA\_CRM\_SelectTransactionType, which you can use without customizing the runtime library.

**Example**

The creation of a sales order is a typical example of a business scenario that starts with the selection of the transaction type. The type is not always at the same position (depending on the number of transaction types in the database). Adapt the test script, so that it does not search for the row number but for the transaction type using the text being displayed.

Note: By default, the line is selected by row number. This information is collected while recording the scenario, and is used in the generated test script.

The following graphic illustrates the automation of this step. The first column shows the transaction type, the second a short description. Both columns can identify the transaction type to be selected.

The bottom of the screen includes a “pager” control to navigate to the next pages if the transaction type is not on the current page.

**Signature of SelectTransactionType Pattern**

* Input parameters
  + TransactionType: text in the **Transaction Type** column
  + Description: (optional) text in the **Description** column
  + Operator: (optional) the operator to use when defining the filter (default is =)

It compares the TransactionType and Description parameter values with the actual values of the respective columns. Supported operators are those documented in the Query API.

* + MaxPage: (optional) the number of pages to search for the transaction type

Default is 10. The implementation navigates to the next page when the transaction type is not in the current one.

* + Options: several options influencing the behavior are available
    - /t = for trimming. Ignores leading and trailing spaces when comparing the TransactionType and Description parameter values with the actual values of the respective columns.
    - /u = upper-case. For a case-insensitive comparison.
* Output parameters
  + Output: report status, OK or FAILED.

SelectTransactionType has 5 parameters to comply with the default component CBTA\_CRM\_A\_INVOKE\_FUNCTION for calling custom functions.

# Pattern for SAP GUI: ME21N\_ExpandAllSections

Restores the initial state of the transaction ME21N/ME51N main screen.

The screen consists of panels. Each panel can be expanded or collapsed, depending on how the transaction ended during the previous session.

Like most of the patterns, the implementation consists of two functions. The first one is a facade to the implementation, which includes statements for handling execution errors. The second method performs the job.

# Pattern for SAP GUI: WaitForCondition

Checks for a condition, and thus stops the execution flow temporarily.

Some SAP GUI applications start a background job to compute business data. The test script may need to pause and wait for the background job to complete, before continuing.

The PATTERN\_NAME fragment is automatically replaced by the final custom function name when using the Code Assistant feature of the runtime library manager:

Code Syntax

Function PATTERN\_NAME ( TransactionType, Description, Operator, MaxPage, Options )

On Error Resume Next ' Important - Exception Handling - Do not change it!

EventComponentBegin()

If ConditionsManager().CheckConditions() Then PATTERN\_NAME =

PATTERN\_NAME\_Impl ( TransactionType, Description, Operator,

MaxPage, Options)

End If

EventComponentEnd()

End Function

Function PATTERN\_NAME\_Impl ( TransactionType, Description, Operator, MaxPage,

Options ) 'internal

ImportLibrary "CRM\_WebController.vbs"

ImportLibrary "CRM\_WebControls.vbs"

PATTERN\_NAME\_Impl = "NOT DONE"

...

End Function

## Initialization Phase

The first statements of the implementation initialize local variables and validate the input parameters:

Code Syntax

Dim query, filter, subFilter, selectorInPopup, maxPageAsInt, i, uriNextPage,

nextPageControl

maxPageAsInt = ToInt(MaxPage) If

IsNull(maxPageAsInt) Or maxPageAsInt = 0 Then

maxPageAsInt = 10 End If

If IsNull(Operator) Then

Operator = "="

End If

If IsNull(Options) Then

Options = ""

End If

ReportDebugLog writes information to the traces for troubleshooting:

Code Syntax

ReportDebugLog "CUSTOM Function - PATTERN\_NAME" & \_

" - Parameters: TransactionType " & TransactionType & " -

Description: " & Description

ReportDebugLog "CUSTOM Function - PATTERN\_NAME" & \_

" - Operator: " & Operator & " - MaxPage: " & maxPageAsInt & " -

Options: " & Options

## Implementation

The implementation consists of a loop (using the FOR/NEXT statements) to search each page for the transaction type.

Within this loop, the Query API defines the row match criteria. The query object is created by the CRM controller. The Query API handles situations that are specific to the underlying UI technology. CRM applications are built on top of the WebCUIF framework. The query created in the example below leverages the meta-information that this framework inserts in the HTML content, so that the test engineer can handle common use cases like this one. For information, see Query API for Testing of Dynamic Scenarios [page 41]

Code Syntax

For i=1 To maxPageAsInt

' Preparation of the CRM Query - Identifying the parent Table container (in the

Popup) Set query = CrmController().CreateQuery()

query.ParentControlUri = "popupId=1;

crm.id=A\_btfollowup:V\_ProcType:T\_cellerator:C\_proctype:I\_"

' Specifying the element type to search for - ROW\_SELECTOR

Set filter = query.SetFilter() filter.AddCondition

"crm.type", "=", "ROW\_SELECTOR" filter.AddCondition

"crm.context", "=", "proctype" If Not

IsNull(TransactionType) Then

' filtering on the Transaction Type

filter.AddCondition "crm.column:proctype.process\_type", Operator,

TransactionType, Options

End If

If Not IsNull(Description) Then ' filtering

on the Transaction Type Description

filter.AddCondition "crm.column:proctype.proc\_type\_descr\_20", Operator,

Description, Options

End If

...

Next

The ParentControlUri property restricts the scope of the query. In this example, the URI of the table (shown in the popup window) is specified.

A filter is defined to specify what to search for: The first condition checks the crm.type attributes. It guarantees that only ROW\_SELECTORmatches the criteria. The second condition checks for the context C\_ fragment of the ID that the WebCUIF framework generates.

Two additional conditions are defined, depending on the input parameters: When the TransactionType parameter is set, an additional condition is created to check for the content of the column with the technical name proctype.process\_type. When the Description parameter is set, another condition is created to check the content of the column with the technical name proctype.proc\_type\_descr\_20.

Note

The query is created within the loop because the query object has a restricted life, which is bound to the HTML document being displayed. When navigating to the next page, the current HTML document is discarded and a new one is displayed, so a new query is necessary.

## Query Resolution

Once defined, the query can be resolved to get the controls that match the criteria. In this use case, we only expect a single control (a single row), so we use the SelectSingle method here. The selectorInPopup variable receives the query result:

Code Syntax

' Selecting the first element matching the criteria

ReportDebugLog " CUSTOM Function - PATTERN\_NAME - now resolving query - Current

Page: #" & i

Set selectorInPopup = query.SelectSingle()

If the transaction type is found, this variable is not NULL, and we check it to determine whether to navigate to the next page.

Code Syntax

If selectorInPopup Is Nothing Then uriNextPage = "label=Navigation to Page #" &

(i+1) & "; " & \_ "popupId=1; tag=SPAN; crm.tag=TD; " & \_

"crm.id=A\_btfollowup:V\_ProcType:T\_PAGER:C\_proctype:I\_:K\_pag\_pg-" &

(i+1)

Set nextPageControl = CrmController().GetCrmControlByUri(uriNextPage)

If nextPageControl Is Nothing Then

ReportLog "FAILED", "PATTERN\_NAME", "Operation Failed - Last page has been

reached" Exit For

Else

' Control not found - Let's navigate to the next page (if any)

CRM\_WebControl\_Click uriNextPage

If ReportStatus <>"OK" Then

Exit For

End If

End If

Else

...

End If

This code builds the URI of the next page and simulates a mouse click to navigate to it.

When the last page is reached, the loop is interrupted and the test script can fail because no transaction type was selected.

## Row Selection

When the row selector UI element is found, the selectorInPopup variable is set. The implementation uses the GetControlUri method to determine the URI of the row selector and calls the CRM\_Table\_SelectRow function to perform the selection.

Code Syntax

' Control found - let's click on the ROW\_SELECTOR

CRM\_Table\_SelectRow selectorInPopup.GetControlUri()

If ReportStatus = "FAILED" Then

ReportDebugLog " CUSTOM Function - PATTERN\_NAME - Operation Failed"

Else

ReportDebugLog " CUSTOM Function - PATTERN\_NAME - Operation Succeeded"

End If

PATTERN\_NAME\_Impl = ReportStatus

Exit For

The PATTERN\_NAME\_Impl variable provides feedback about the status of the operation. It is the value received by the output parameter of theCBTA\_CRM\_A\_INVOKE\_FUNCTION component. Result: The transaction type is selected. The Exit For statement interrupts the loop to avoid searching the next pages.

1. Pattern for SAP GUI: ME21N\_ExpandAllSections

The ME21N\_ExpandAllSections pattern checks for the state of the header sections and expands them automatically. It restores a recognized initial state if a transaction starts in a different state, depending, for example, on user preferences, to let the test script continue in the expected context.

**Example**

This problem typically occurs with transaction ME21N, where the main screen consists of several panels. The initial state of the panels is unknown at startup. Each panel can be expanded or collapsed, depending on how the transaction ended during the previous session. The test script needs to expand this panel before it continues. The following graphics shows the Overview panel, once in collapsed and once in expanded state.

The custom code of this pattern verifies the state of each panel and expands it, if necessary.Note

This situation can usually be resolved using default components such as CBTA\_A\_SETCONDITION, but not with transaction ME21N, which has a very specific handling of its panels. It uses several subscreens to display the content of each panel, so a custom function is necessary.

**Signature of SelectTransactionType Pattern**

* Input parameters: none
* Output parameters: none

We assume that the custom function has been created previously, using the Code Assistant feature of the runtime library manager.

Test script definition: This custom function has no input parameters, so it can be called up by using the CBTA\_GUI\_A\_EXECUTESTATEMENT default component:

Input parameters: The component for calling the custom function expects at least the Library parameter (name of the VB script file) and theStatement parameter (statement to execute). If the custom function does not expect input parameters, the statement is simple.

Execution report: The actions performed by the custom function are displayed in the execution report.

The implementation consists of sub-procedures. The first one is a facade of the implementation. The second method performs the job.

The PATTERN\_NAME fragment is automatically replaced by the final custom code name when using the Code Assistant feature of the runtime library manager.

Code Syntax

Sub PATTERN\_NAME () On Error Resume Next ' Important - Exception Handling for

SAPGUI - Do not change it! EventComponentBegin()

If ConditionsManager().CheckConditions() Then

If ConditionsManager().CheckConditions() Then

End If

EventComponentEnd()

End Sub

Sub PATTERN\_NAME\_Impl () 'internal

ReportDebugLog "CUSTOM Function - PATTERN\_NAME"

...

End Sub

The subprocedure calls the Expand\_Impl function for each collapsible section in the ME21N transaction.

Code Syntax

Uri = "label=Expand Header; type=GuiButton;

id=wnd[0]/usr/subSUB0:SAPLMEGUI:0016/subSUB1:SAPLMEVIEWS:1100/

subSUB1:SAPLMEVIEWS:4000/btnDYN\_4000-BUTTON" Expand\_Impl Uri

Uri = "label=Expand Item Overview; type=GuiButton;

id=wnd[0]/usr/subSUB0:SAPLMEGUI:0016/subSUB2:SAPLMEVIEWS:1100/

subSUB1:SAPLMEVIEWS:4001/btnDYN\_4000-BUTTON"

Code Syntax

Expand\_Impl Uri Uri = "label=Expand Item Detail; type=GuiButton;

id=wnd[0]/usr/subSUB0:SAPLMEGUI:0016/subSUB3:SAPLMEVIEWS:1100/

subSUB1:SAPLMEVIEWS:4002/btnDYN\_4000- BUTTON" Expand\_Impl Uri

The Expand\_Impl procedure is also delivered with the pattern code. It searches for the collapsible button by URI, calling theFindGuiControlByUri method. This call returns a reference to a GuiScripting object (a GuiButton in this case). The code then determines whether the panel is expanded or collapsed, using the IconName property.Note

The logic is specific to the ME21N transaction. Checking the IconName is not the regular way of determining the collapsible state, but it is the only information available in the ME21N transaction.

Code Syntax

Sub Expand\_Impl ( Uri ) 'internal

Set GS\_guiControl = Nothing On Error Resume Next

' Switching ON a local error handling

Set GS\_guiControl = GuiScripting().FindGuiControlByUri(Uri)

On Error Goto 0 ' Switching OFF the local error handling

If GS\_guiControl Is Nothing Then ' Control not found ReportLog

"INFO", "Expand", "Operation skipped - the button does not exists"

Else

If GS\_guiControl.IconName = "DAARSO" Then ReportLog "INFO",

"Expand", "Operation skipped - the section is already expanded" Else

GS\_guiControl.Press ' Control has been found - let's click on it

ReportLog "INFO", "Expand", "Operation has been performed" \_

& vbCrlf & "Target: " & GetMeaningfulName(GS\_guiControl)

End If

End If

End Sub

# Pattern for SAP GUI: ME51N\_ExpandAllSections

ME51N\_ExpandAllSections is analog to ME21N\_ExpandAllSections. The only difference is that it expands ME51N collapsible panels.

Pattern for SAP GUI: WaitForCondition

The WaitForCondition pattern makes a test script wait for the background process to complete before continuing. With SAP GUI, some transactions can start asynchronous background processes on the server, so the UI cannot display the operation result automatically. The end user must perform actions (like pressing ENTER) to update the UI with the server status.

The pattern performs an action several times until a text is displayed in the SAP GUI UI.

Signature of WaitForCondition Pattern

* Input parameters
  + Uri: URI identifying the SAP GUI control displaying the operation result

It displays the text for which the wait condition checks. If empty, the URI of the status bar is used by default. Default value: label=Main Window; type=GuiMainWindow; id=/app/con[0]/ses[0]/ wnd[0]

* + TextToCheck: text or part of the text that the control displays when the background process is still running

Default value: Scheduled

* + WaitTime: time in milliseconds to wait between attempts

Default: 1000 (one second)

* + MaxAttempts: maximum number of attempts

When the maximum number is reached, the implementation stops checking and exits with NOT\_DONE as output parameter. Default : 30.

* + Options: not used
* Output parameters:
  + Output: status, NOT\_DONE or DONE

The pattern has 5 parameters, so it can be called by the CBTA\_GUI\_A\_INVOKE\_FUNCTION default component.

Like most of the patterns, the implementation consists of two functions. The first one is a facade of the implementation, which includes execution error handling statements. The second method performs the job.

**Initialization Phase**

The first statements validate the input parameters and set their default values, if necessary.

Code Syntax

If IsNull(Uri) Then

Uri = "label=Status Bar; type=GuiStatusbar; id=/app/con[0]/ses[0]/wnd[0]/sbar"

End If

If IsNull(TextToCheck) Then

TextToCheck = "Scheduled"

End If

If IsNull(WaitTime) Then

WaitTime = "1000" '1 second

End If

If IsNull(MaxAttempts) Then MaxAttempts = "30" '30 \* 1000 ==> 30 seconds max!

End If

The code writes information to the traces, for troubleshooting, and initializes the result to NOT DONE.

Code Syntax

ReportDebugLog "CUSTOM Function - PATTERN\_NAME" & \_

vbCrLf & " - Uri: " & Uri & \_ vbCrLf & " -

TextToCheck: " & TextToCheck & \_ vbCrLf & "

- WaitTime " & WaitTime & \_ vbCrLf & " -

MaxAttempts: " & MaxAttempts

PATTERN\_NAME\_Impl = "NOT DONE"

**Implementation**

A loop, using the For/Next keywords, checks several times for the text being displayed, until the limit is reached, which is defined by themaxAttempts input parameter.

The GetPropertyValue method retrieves the text that is displayed by the control to which the URI refers. It then checks if this text contains the value in the TextToCheck input parameter: If so, it waits for the defined time and executes ENTER to refresh the screen. If not, it assumes the background process is over and breaks the iteration.

Code Syntax

Dim i, statusBarUri, mainWindowUri, statusBarText, max mainWindowUri =

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

max = ToInt(MaxAttempts) For i=1 To max ReportDebugLog

"PATTERN\_NAME - Now getting the text of the Control" statusBarText

= GetPropertyValue ( Uri, "Text", Null ) ReportDebugLog

"PATTERN\_NAME - Control Text: " & statusBarText

If InStr(statusBarText, TextToCheck) Then

' Let's wait ReportLog "INFO", "PATTERN\_NAME", "Waiting for

background job to complete - attempt #" & i GS\_Wait WaitTime, Null

'Let's Press Enter to refresh the UI

GS\_PressKey mainWindowUri, "Enter", Null

Else

' Text not visible anymore

ReportLog "DONE", "PATTERN\_NAME", "Background job has completed"

PATTERN\_NAME\_Impl = "DONE"

'Let's stop the loop

Exit For

End If

Next

# Patterns Location

The patterns that are predefined by SAP are stored under MIME Repository (SE80) of the SAP Solution Manager system, under SAP  PUBLIC  CBTA  PATTERNS. Patterns are stored in sub-folders, depending on their nature

# Runtime Library Manager

The runtime library manager customizes the runtime library. It has the following capabilities:

* Opening the runtime library for editing
* Writing custom code manually
* Writing custom code by using the patterns with the code assistant
* Testing the custom code before submitting it
* Submitting changes to make them available to all testers

## Code Assistant

When you use the code assistant, a guided procedure helps you to select a pattern and generate the custom code.

Selecting a Pattern

Select a pattern before you create a custom function. If you use the assistant instead of doing it manually, the generated code will be ready for use. Select a pattern according to your needs or the problem that you are facing. The UI technology of the application that is tested is the main criterion for the pattern selection.

Generating a Custom Function

The code assistant finishes with the generation of the custom code, using the function name that the test engineer specified.

The VB script containing the custom code is stored locally on the file system, at the location specified by the test engineer. Adapt the VB script to your needs, and test it, before submitting the changes.

# More Information

* SAP Note [1778899 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1778899)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1778899) (CBTA collective note, including release documentation on the SPs)
* SAP Note [1912801 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1912801)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1912801) (CBTA - RTL Manager - Runtime Library Customization)
* Scripting Objects Interfaces from Microsoft

The MSHTML API is a set of Microsoft COM interfaces that provides access to the HTML content that MS Internet Explorer displays. The official documentation is online on MSDN at [https://msdn.microsoft.com/en-us/library/aa741322(v=vs.85).aspx[Information published on non-SAP site](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/aa741322(v=vs.85).aspx)](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/aa741322(v=vs.85).aspx)

| **Interface Name** | **Method Name** | **Link** |
| --- | --- | --- |
| IHTMLElement | click | [https://msdn.microsoft.com/en-us/library/%20aa752277(v=vs.85).aspx[Information published on non-SAP site](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa752277(v=vs.85).aspx)](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa752277(v=vs.85).aspx) |
| IHTMLElement | all | [https://msdn.microsoft.com/en-us/library/%20aa752281(v=vs.85).aspx[Information published on non-SAP site](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa752281(v=vs.85).aspx)](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa752281(v=vs.85).aspx) |
| IHTMLElementCollection | tags | [https://msdn.microsoft.com/en-us/library/%20aa703932(v=vs.85).aspx[Information published on non-SAP site](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa703932(v=vs.85).aspx)](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa703932(v=vs.85).aspx) |
| IHTMLElement3 | fireEvent | [https://msdn.microsoft.com/en-us/library/%20aa703947(v=vs.85).aspx[Information published on non-SAP site](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa703947(v=vs.85).aspx)](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa703947(v=vs.85).aspx) |
| IHTMLElement3 | setActive | [https://msdn.microsoft.com/en-us/library/%20aa703969(v=vs.85).aspx[Information published on non-SAP site](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa703969(v=vs.85).aspx)](http://help.sap.com/disclaimer?site=https://msdn.microsoft.com/en-us/library/%20aa703969(v=vs.85).aspx) |

# Query API for Testing of Dynamic Scenarios

The testing of some scenarios cannot be automated by the regular approach that is based on components.

Like most applications, the Web applications that SAP delivers display their content in Microsoft Internet Explorer as HTML pages. The HTML content can be very complex and dynamic. It is generated on the server by the presentation layer, which can differ depending on the UI technology used to develop the application.

Main UI technologies at SAP

* SAPUI5 / FIORI
* WebCUIF

Layer that generates the HTML content for SAP CRM Web UI applications.

* Unified Rendering Light Speed (LS)

Layer that generates the HTML content for SAP Applications such as:

* + Web Dynpro Application (ABAP)
  + Web Dynpro Applications (Java – recent versions)
  + SAP GUI for HTML– SAP GUI content displayed in browser
* BSP
* Applications based on Business Server Pages.
* Java Web Dynpro

Layer formerly used by Java Web Dynpro applications.

Non-SAP UI technologies

* Plain HTML pages (supported in CBTA)

Content generated with regular HTML tags (with no, or few, scripting capabilities).

* Adobe Flash (not supported in CBTA)

Layer used by Adobe Flash applications, and generated using the Adobe Flex software development kit.

* Java Applets (not supported in CBTA)

Application written in Java. Java applets can appear in a frame of the Web page, or in a new application window.

Each UI technology has its own logic and generates HTML content that can be completely different from others. This is challenging for test automation, and can require custom functions. In custom functions, the test engineer can search the HTML content displayed by the application for UI element, using native COM interfaces (from Microsoft) that the MS Internet Explorer object exposes. However, this approach is complex and error-prone. The alternative to the native COM interfaces is using queries via an API.

# Structure of the Query API

The query API hides the HTML complexity and searches for UI elements with filters and conditions.

## Query Types

The query approach is similar, regardless of the underlying technology, but some additional capabilities can be available for some UI technologies. The queries are application-specific. The following table shows the query types and their UI technologies.

| **Query Type** | **UI Technology** | **Applications** |
| --- | --- | --- |
| CRM Queries | WebCUIF | SAP CRM applications |
| LS Queries | Unified Rendering Light Speed (LS) | Web Dynpro application (ABAP)  Web Dynpro applications (Java)  Web GUI – SAP GUI content displayed in  MS Internet Explorer |
| Web Queries | Unified Rendering Light Speed (LS) | Web applications |
| JWDP Queries | Java Web Dynpro (old releases) | Java Web Dynpro applications |

Example query: "Search for all UI elements in an HTML table in which the Partner Function is Sales employee and the Partner ID is 244."

## Application Controllers

Application controllers are the main entry point to the application being tested, and provide access to the HTML content that the current session displays.

Several application controllers are provided, each for a specific use-case, depending on the underlying UI technology:

* Light Speed Controller: Controller for applications that are built on top of the Unified Rendering Light Speed framework (such as Web Dynpro ABAP applications).
* CRM Controller: Controller for SAP CRM applications that rely on the WebCUIF UI technology.
* Web Controller: Controller for Web applications whose underlying UI technology is unknown, or Web applications that do not have a specific controller (like BSP applications).
* Java Web Dynpro Controller: Controller for applications based on previous releases of Java Web Dynpro.

The controller interfaces are the entry points to the query interfaces. Each controller has at least one method to create query objects.

Creating a Web Query Object: The following script creates a query object to search for Web controls:

Code Syntax

Dim controller, query

Set controller = WebController()

Set query = controller.CreateQuery()

Creating a WebCUIF query object: The following script creates a query object to search for WebCUIF controls (for SAP CRM applications), usingtheCrmController.

Code Syntax

Dim controller, query

Set controller = CrmController()

Set query = controller.CreateQuery()

Creating a Light Speed query object: The following script creates a query object to search for Light Speed controls. The code is similar; the only difference is that the Light Speed controller is used instead of the Web controller.

Code Syntax

Dim controller, query

Set controller = LsController()

Set query = controller.CreateQuery()

Creating a Web query object from any controller: In the following example, we ask for a Web query instead of a Light Speed query, explicitly, by calling theCreateWebQuery method.

Code Syntax

Dim controller, query

Set controller = LsController()

Set query = controller.CreateWebQuery()

Query Creations with Options: The query objects can be created with options that are specific to the underlying UI technology. This additional parameter is optional. The following example shows the creation of a Light Speed query using the option /ls, which activates optimization. For more details, see the following sections.

Code Syntax

Dim query

Set query = LsController().CreateQuery("/ls")

## Web Controls

Light Speed Web Controls

Applications that are built on top of the Light Speed framework generate controls in which at least one of the HTML elements provides meta-information by additional HTML attributes:

* ct: control type
* lsdata: Light Speed data; a collection of key/value pairs that describe the control and its current state.

Example of Light Speed attributes for a control of type checkbox:

* Checked: state of the checkbox
* Enabled: false when the checkbox is disabled (its state is not relevant)
* Readonly: false when the state cannot be changed
* Text: the text displayed next to the checkbox
* Tooltip: the text in the tooltip

The Light Speed query interfaces are specialized to use this meta-information, so you can search for a control using the Light Speed data.

CRM Web controls

CRM applications generate their content with the WebCUIF UI technology, in which each control has a hidden SPAN HTML element that provides meta-information by its ID attribute. Such an ID can look like this, for example:

A\_btpartner:V\_Partner:T\_ROW\_SELECTOR:C\_btpartner:I\_partner\_fct:R\_1

The ID consists of the following elements:

* A\_: application name
* V\_: view name
* T\_: type of the control
* C\_: context
* I\_: interface, which is a field name or an attribute name in the C\_ context
* R\_: row number; for example, when the control embedded in a table container
* K\_: key; for example, an item in a dropdown list box can be selected by key

The CRM query interfaces are specialized to use this meta-information, so you can search for a control with these fragments.

## Example

The screenshot below shows the kind of toolbar that a CRM application can display. In this example, the links are illustrated with an icon:

......................................................................

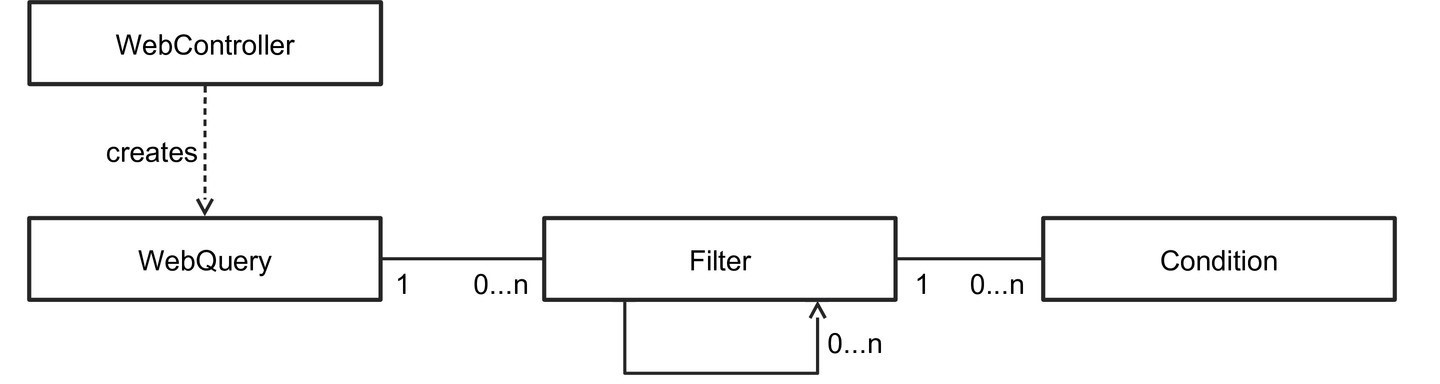
You can use the developer tool (F12 in MS Internet Explorer) to visualize the internal structure of the Save button. The result will show the following HTML elements:

......................................................................

Even for this simple toolbar, the internal structure is complex. A lot of information is available, and the query API hides this multiplicity. With the query API, the test engineers can, for example, search the Save button using the text which is visible to the end user, which is in the innerTextHTML property.

## Queries, Filters and Conditions

The query API searches controls in HTML pages. The objects created with this API have the following characteristics:

* Queries are created with an application controller. Select the one for the UI technology that is tested.
* The query can have one or more filters.
* Each filter can have one or more conditions.
* Each filter can have sub-filters.
* Each condition can check the values of the UI element attributes and properties.

<Figure 1: The query class diagram illustrates the relationship between the application controller, the queries, the filters and the conditions.>

Queries have a restricted lifecycle. They can only be used in the same context in which they are created. A query can be used several times, but only for the current set of HTML documents, so you need to create a new query when you navigate to a different HTML page, that is, after each roundtrip to the HTTP server.

## Interface Overview

Controller Interface

The controller is the entry point to the query API. You can create queries with it.

With the CreateQuery method, you can create objects for the query.

There is one controller for each UI technology. Select the one for the element you are looking for.

Query Interface

The main methods of the query interface are the following:

* SetFilter: removes all filters and defines the first one
* AddFilter: adds an additional filter
* Select: executes the query and returns a set of controls
* SelectSingle: executes the query and returns the first control matching the criteria

The query property ParentControlUri defines the scope in which you search for controls.

You can specify the query interfaces to leverage the meta-information for the underlying UI technology.

## Filter and Conditions Interface

The filter contains the criteria that the controls must meet. Methods:

* SetCondition: removes all conditions, and defines the first one
* AddCondition: adds an additional condition
* SetFilter: removes all sub-filters and defines the first one
* AddFilter: adds an additional sub-filter

A condition contains information about the criteria that were defined before, using the SetCondition method.

Properties:

* AttributeName: name of the attribute (or property) used by this condition
* BooleanOperator: operator used when evaluating this condition
* Value: expected value of the attribute or property
* Option: evaluation options for condition

## Web Control Interfaces

A query searches the HTML content for controls, and returns one or more of them. These controls implement interfaces, providing access to the HTML attributes and properties of the underlying HTML elements.

A single control can be a combination of several HTML elements, and the API is specialized to provide access to the meta-information by additional interfaces, such as:

# Examples for Using the Query API

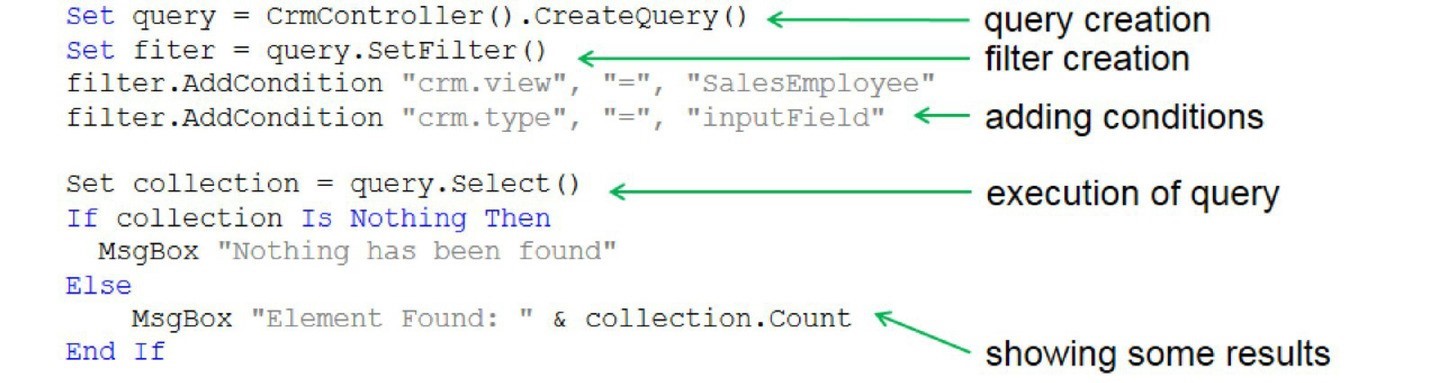
## Example: Contact Creation

Three views are used to enter the contact information:

* bp\_cont/ContactDetails
* bp\_cont/SalesEmployee
* bp\_addr/StandardAddress

The query API can retrieve all the input fields in the SalesEmployee view.

Searching for Web Controls

The following script shows how to create and execute a query:

The VB Script accesses the HTML document by calling the CrmController method, and creates a query using the CreateQuery method. The query is empty. Specify the criteria to identify the controls we are looking for, using filters and conditions. In our example, a single filter with two conditions is specified. The filter is created using the SetFilter method, and the two conditions are added using the AddCondition method.

Once the criteria are set, the query can be executed with the following methods:

* Select: returns a collection of CRM Web controls which match the criteria.
* SelectSingle: returns the first CRM Web control which matches the criteria, not a collection

The next exemplary sections describe how to use the results of the Select and SelectSingle methods.

Setting the Input Field Value

Our first example explains how to create and execute a query. Our second example will show how to use the result, and manipulate the retrieved CRM Web control object.

Code Syntax

Dim controller, query, filter, crmControl, controlUri

Set controller = CrmController()

Set query = controller.CreateQuery() Set filter =

query.SetFilter() filter.AddCondition "crm.view",

"=", "SalesEmployee" filter.AddCondition

"crm.type", "=", "inputField" Set crmControl =

query.SelectSingle() If Not crmControl Is Nothing

Then controlUri = crmControl.GetControlUri()

controller.SetElementValue controlUri, "1234"

End If

This script invokes the SelectSingle method to retrieve a single CRM Web control (so there is no need to iterate through a collection). It checks whether the crmControl variable is set, and asks for the URI identifying the CRM Web control, by calling the GetControlUri method. This call returns a URI which complies with those that the runtime library expects when you use default components.

In this example, this is the URI that identifies the CRM Web control:

crm.area=WorkArea; tag=INPUT;

crm.id=A\_bp\_cont:V\_SalesEmployee:T\_inputField:C\_salesemployee:I\_struct.salesemployee

With this URI, the script can invoke one of the methods exposed by the CRM Web controller API to perform actions that are normally performed with default components. In our example, the script sets the value of the input field to 1234, by calling the SetElementValue method.

## Example: Appointment Creation

The scenarios that we have seen so far are not dynamic, so the query API is not necessary, because the default components are normally sufficient to automate them. The next example explains how to select CRM Web controls by filtering on the values of the INPUT HTML element that the browser displays.

Searching for Web Controls in a Table

We must automate a dynamic scenario in which the script selects the first row in which the partner function is Sales employee and Partner ID is 244.

The difficulty is that the conditions are to be checked against the CELL contents in the same row but a different column, not against the CRM Web control that is being searched. This kind of query can be very complex when it searches the internal hierarchy of HTML elements. To make it simple, the filters and the conditions have been specified to hide the complexity and handle common use cases like this one.

This is what the VB Script coding for row selection looks like:

Code Syntax

Dim controller, query, filter, selector, controlUri

Set controller = CrmController()

Set query = controller.CreateQuery() Set filter = query.SetFilter()

filter.AddCondition "crm.application", "=", "btpartner"

filter.AddCondition "crm.view", "=", "Partner" filter.AddCondition

"crm.context", "=", "btpartner" filter.AddCondition "crm.type", "=",

"ROW\_SELECTOR" filter.AddCondition "crm.column:btpartner.partner\_no",

"=", "244" filter.AddCondition "crm.column~Partner Function", "=",

"Sales employee" Set selector = query.SelectSingle()

If Not selector Is Nothing Then

If Not selector Is Nothing Then

If Not selector Is Nothing Then

End If

This script shows the types of conditions that the query supports. The conditions here check the attributes of the CRM Web controls. The API lets you check the regular HTML attributes and properties, and some CRM-specific attributes to check the fragments of the CRM IDs.

The object spy (or the developer tool from MS Internet Explorer) shows the CRM ID of the first row selector:

A\_btpartner:V\_Partner:T\_ROW\_SELECTOR:C\_btpartner:I\_:R\_1

The fragments provide meta-information about the CRM Web control:

* A\_: application
* V\_: view name
* T\_: type
* C\_: context node
* I\_: interface; for input fields, the technical name
* R\_: row number

The condition can check the value of these fragments using CRM attributes like the following:

* crm.application: checks the application
* crm.application: checks the view name
* crm.application: checks for the UI element type

The following filtering guarantees that the query will only return the row selectors:

Code Syntax

filter.AddCondition "crm.application", "=",

"btpartner" filter.AddCondition "crm.view", "=",

"Partner" filter.AddCondition "crm.context", "=",

"btpartner" filter.AddCondition "crm.context", "=",

"btpartner"

We now need to select the second one only (the second visible line). If our scenario was not dynamic, we could select the second row directly, using a default component (for example SelectRow) and specify the URI:

A\_btpartner:V\_Partner:T\_ROW\_SELECTOR:C\_btpartner:I\_:R\_2

Because our scenario is dynamic, the row we want to select may not always be in the second position. So we need to search for it by checking the value of the other columns. Two syntaxes are possible when you filter for column values, depending on whether you pass the column title or the column technical name.

The syntax can be crm.colum:<context>.<interface> or crm.colum~<Column Title>, so the following two ways of filtering are equivalent:

Code Syntax

filter.AddCondition "crm.column:btpartner.partner\_no", "=", "244"

filter.AddCondition "crm.column:btpartner.partner\_fct ", "=", "Sales

employee"

...

Code Syntax

filter.AddCondition "crm.column~Partner ID", "=", "244"

filter.AddCondition "crm.column~Partner Function", "=", "Sales

employee"

In our example, both syntaxes are used:

Code Syntax

filter.AddCondition "crm.column:btpartner.partner\_no", "=", "244"

filter.AddCondition "crm.column~Partner Function", "=", "Sales

employee"

## Tips and Tricks

Iterating Through a Set of Web Controls

The following code shows how to iterate though the set of CRM Web controls that are retrieved by the query. The collection implements the Web Control Collection Interface described in [Technical Information for the Query API](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0b6f4d347093480cb24d5734d2691782.html).

Code Syntax

Dim collection

Set collection = query.Select()

If collection Is Nothing Then

MsgBox "Nothing has been found"

Else

MsgBox "Element Found: " & collection.Count

Dim childControl

for i=0 To collection.Count-1 Set

childControl = collection.ControlAt(i)

MsgBox "Uri: " & childControl.GetControlUri()

Next

End If

Note

The VB Script For Each statement is not supported.

Making Information Available to Subsequent Components

Usually, only a part of the tested scenario is dynamic and requires custom coding and queries. Then it can be useful to make the URI of the retrieved CRM Web controls available to subsequent components, and continue the scenario using default components. This is typically done with the SetToken method, as shown in the following example:

Code Syntax

Dim crmControl, crmControlUri

Set crmControl = query.SelectSingle()

If crmControl Is Nothing Then CBTA.Log "Unexpected

Situation - Control not found" Else

crmControlUri = crmControl.GetControlUri() CBTA.Log

"Uri: " & crmControlUri ExecutionContext.SetToken

"rowSelector", crmControlUri

End If

Subsequent components can reuse the value that is stored in the execution context, using the %token% syntax. In this example, the subsequent components can use %rowSelector% to perform operations on the ROW\_SELECTOR that was selected by the query.

Writing Information to the Execution Report

The CBTA object provides the following methods:

* CBTA.Log: troubleshoots complex scenarios

Public Sub Log(message)

* CBTA.Report: provides feedback to the tester by the Execution Report

Public Sub Report(Severity, Topic, Message, Options)

Reducing the Scope of a Query

By default, the CRM queries automatically search for CRM Web controls in the FRAME work area of the main browser window. The query analyzes the content of the body of the HTML document, and checks the filters and conditions for all CRM Web controls that were found.

For performance reasons, you can reduce the scope of the query and specify where to start searching. For instance, when searching for CRM Web controls of type ROW\_SELECTOR, first specify the parent container HTML table element, using the ParentControlUri property of the query. For example:

Code Syntax

'--------------------------------------------

' Query Example with Restricted Scope '--------------------------------------------

Dim query, filter, subFilter Set query =

CrmController().CreateQuery() query.ParentControlUri =

"tag=DIV; crm.id=A\_btpartner:V\_Partner:T\_cellerator:C\_btpartner:I\_"

Note

The URI should be determined by the object spy, but the HTML element table cannot always be selected, depending on the style.

The alternative to the object spy is to search for the CRM ID of the cellerator UI element with the developer tool in MS Internet Explorer.

Searching for Web Controls in a Popup Window

The queries search for CRM Web controls in the FRAME work area of the main browser window, by default. You can change this default behavior with the ParentControlUri property.

Example VB Script to search for a control in a modal popup window:

Code Syntax

Dim controller, query, filter

Set controller = CrmController()

Set query =

controller.CreateQuery()

query.ParentControlUri = "popupId=1; crm.area=WorkArea;

tag=body" Set filter = query.SetFilter() filter.AddCondition

"crm.type", "=", "ROW\_SELECTOR" filter.AddCondition

"crm.context", "=", "proctype" filter.AddCondition

"crm.context", "=", "proctype"

filter.AddCondition "crm.column:proctype.proc\_type\_descr\_20", "=",

"Meeting" Dim selectorInPopup, selectorUri Set selectorInPopup =

query.SelectSingle() selectorInPopup.HighLight "red" selectorUri =

selectorInPopup.GetControlUri() controller.SearchResult\_SelectRow

selectorUri

This example contains nothing new regarding the selection of the ROW\_SELECTOR. The script uses some conditions to select the transaction with type 1001 and type description Meeting. The difference is that it now specifies searching in the FRAME work area of the popup by setting the ParentControlUri property:

query.ParentControlUri = "popupId=1; crm.area=WorkArea; tag=body"

In this example, it would have been possible to restrict the scope of the query further, by specifying the URI of the TABLE HTML element like this:

query.ParentControlUri = "popupId=1;

crm.id=A\_btfollowup:V\_ProcType:T\_cellerator:C\_proctype:I\_"

Combining Filters for Complex Queries

All examples so far use only one filter with several conditions. Conditions are evaluated one by one, and the CRM Web UI elements are excluded as soon as a condition is not met. The conditions are connected by a logical AND operator.

Our previous example selected the first row with the following conditions: Partner Function equals "Sales employee" AND Partner ID equals "244".

Instead of searching for the ROW\_SELECTOR, we could retrieve controls that are either checkboxes or input fields that are associated with an F4help, by combining several filters, as in the following example:

Code Syntax

Dim query, filter, subFilter, collection, childControl

Set query = CrmController().CreateQuery()

query.ParentControlUri = "tag=DIV;

crm.id=A\_btpartner:V\_Partner:T\_cellerator:C\_btpartner:I\_" Set filter =

query.SetFilter() filter.AddCondition "crm.application", "=",

"btpartner" filter.AddCondition "crm.view", "=", "Partner"

filter.AddCondition "crm.context", "=", "btpartner" filter.AddCondition

"crm.column:btpartner.partner\_no", "=", "244" filter.AddCondition

"crm.column~Partner Function", "=", "Sales employee" Set subFilter =

filter.SetFilter()

subfilter.AddCondition "crm.tagType", "=", "valueHelp"

Set subFilter = filter.AddFilter()

subfilter.AddCondition "crm.tagType", "=", "checkbox"

Set collection = query.Select()

If collection Is Nothing Then

CBTA.Report CBTA.FAILED, "Demo Query", "No controls have been found", ""

Else CBTA.Report CBTA.INFO, "Demo Query", "Nb. Controls: " &

collection.Count, "" for i=0 To collection.Count-1 Set childControl =

collection.ControlAt(i)

CBTA.Report CBTA.INFO, "Demo Query", "Uri: " & childControl.GetControlUri(), ""

Next

End If

In this example, the main filter no longer checks for the ROW\_SELECTOR type, so all cells in the row match its criteria. The difference is that we have now defined two sub-filters that specify the type of the UI elements for which we are looking.

The first sub-filter checks for elements that are associated with an F4 help, which use the valueHelp tagType.

Code Syntax

Set subFilter = filter.SetFilter()

subfilter.AddCondition "crm.tagType", "=", "valueHelp"

The second sub-filter checks for checkbox UI elements:

Code Syntax

Set subFilter = filter.AddFilter()

subfilter.AddCondition "crm.tagType", "=", "checkbox"

There are two sub-filters here. Both have a single condition which checks for the crm.tagType attribute.

* The first sub-filter is created by calling the SetFilter method on the main filter object.
* The second sub-filter is added by the AddFilter method.

By defining several filters, we specify that we are looking for cells that match both CRM tag types. This is possible because, unlike conditions, the filters are connected by the logical OR operator.

Reusing Default Component Implementations

The controller interfaces also expose methods to interact with the application UI. There is at least one method per default component that CBTA delivers.

The two examples below are equivalent:

Code Syntax

CRM\_Table\_SelectRow crmControl.GetControlUri()

If ReportStatus = "FAILED" The

' Nothing to do

End If

...

Code Syntax

Set operationResult = CrmController().Table\_SelectRow crmControl.GetControlUri()

If operationResult.Status = "FAILED" Then

CBTA.Report CBTA.FAILED, "Demo Query", "Row selection failed", ""

Else

CBTA.Report CBTA.INFO, "Demo Query", "Row has been selected", ""

End If

The first example invokes the component implementation, while the second one invokes the corresponding method via the controller interface. The advantage of the first approach is that it benefits from default exception handling (writing information in the execution report when an error occurs). In the second example, the script is responsible for checking whether the operation succeeds.

Note

In general, the public methods of the controller interfaces return an Operation Result Object that implement the Operation Result Interface(see [Technical Information for the Query API](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0b6f4d347093480cb24d5734d2691782.html)). The information that is available there can provide human-readable feedback in the execution report.

See Writing Information to the Execution Report, earlier in this document.

Mapping Between Public Methods and Component Implementations Execution Report

The following list shows examples of methods, which are exposed by the controller interfaces, and their component implementations:

| **Controller** | **Method Name** | **Component Implementation (VB Script Function)** |
| --- | --- | --- |
| CrmController() | CaptureScreen | SetElementValue |
| WebControl\_Click | CRM\_WebControl\_Click |
| SetElementValue | N/A |
| **Controller** | **Method Name** | **Component Implementation (VB Script Function)** |
| WebController() | CaptureScreen | WEB\_CaptureScreen |
| WebControl\_SetValue | WEB\_WebControl\_SetValue |
| WebControl\_CheckProperty | WEB\_WebControl\_CheckPropert |

## Example: Finding a Row by Content

CBTA contains default components that search tables for a row, by the content of one of the cells. The following components are available:

* CBTA\_LS\_T\_FindRow: searches a Light Speed table for a row, by checking the value of a single cell.

Input parameters:

* + URI: specifies the URI of the table container
  + ColumnTitle: title of the column (the visible text)
  + Operator: boolean operator used to compare the actual value with the expected one
  + Content: expected value of the cell
  + Options: Some options can be used to convert the values before comparing them.

Output parameters: The number of the row matching the criteria.

* CBTA\_CRM\_T\_FindRow: similar to CBTA\_LS\_T\_FindRow but for SAP CRM Web applications

Both components were implemented using the query API, so a test engineer can copy and paste the source code of the component implementation, and adapt it to his needs.

Note

The code of the CBTA\_LS\_T\_FindRow is in the runtime library, it is visible in the LS\_Functions.vbs file.

Use the runtime library manager to customize the runtime library and write custom functions. When you open the runtime library for editing, the LS\_Functions.vbs file is made available, locally in the file system, at the location that has been specified by the test engineer.

VB Script Function – LsTable\_FindRowByContent

Code Syntax

Public Function LsTable\_FindRowByContent ( Uri, ColumnTitle, Operator, Content,

Options )

On Error Resume Next

EventWebComponentBegin()

LsTable\_FindRowByContent = LsTable\_FindRowByContent\_Impl( Uri, ColumnTitle,

Operator, Content, Options )

EventWebComponentEnd()

End Function

Public Function LsTable\_FindRowByContent\_Impl ( Uri, ColumnTitle, Operator,

Content, Options )

LsTable\_FindRowByContent\_Impl = ""

If IsNull(Uri) Or IsNull(ColumnTitle) Or IsNull(Content) Then

Exit Function

End If

If IsNull(Operator) Then

Operator = "="

End If

If IsNull(Options) Then

Options = ""

End I ReportDebugLog "LsTable\_FindRowByContent" & \_

vbCrLf & "Uri: " & Uri & \_ vbCrLf &

"ColumnTitle: " & ColumnTitle & \_ vbCrLf

& "Operator: " & Operator & \_ vbCrLf &

"Content: " & Content & \_ vbCrLf &

"Options: " & Option Dim controller, query, filter

Set controller = LsController() Set

query = controller.CreateLsQuery()

query.ParentControlUri = uri Set filter =

query.SetFilter() filter.AddCondition

"tag", "=", "TD" filter.AddCondition

"ls.subtype", "=", "SC"

filter.AddCondition "ls.column~" & ColumnTitle, Operator, Content, Options

Set control = query.SelectSingle()

If control Is Nothing Then ReportLog "FAILED", "LsTable\_FindRowByContent",

"Operation Failed - Row could not be found" Else

LsTable\_FindRowByContent\_Impl = control.GetRelevantControl().GetLsRow()

If InStr(Options, "/Select" ) Then

ReportLog "INFO", "LsTable\_FindRowByContent", \_

"Operation succeeded - Row has been selected" & vbCRLf & \_

"Row is: " & LsTable\_FindRowByContent ReportDebugLog

"LsTable\_FindRowByContent - Row Selector Uri: " &

control.GetControlUri() WEB\_WebControl\_Table\_SelectRow

control.GetControlUri()

Else

ReportLog "INFO", "LsTable\_FindRowByContent", \_

"Operation succeeded - Row is: " & LsTable\_FindRowByContent

End If

End If

End Function

## Component LsTable\_FindRowByContent

The following sections explain in detail how this component has been implemented.

Exception handling: The implementation consists of two functions. The first function is required only because the VB Script syntax is not good for exception handling and the On Error Resume Next keyword used here is the only available option.

All default components are built according to the same concept; the first surrounds the invocation of the actual implementation to intercept any errors.

Code Syntax

Public Function LsTable\_FindRowByContent ( Uri, ColumnTitle, Operator, Content,

Options )

On Error Resume Next

EventWebComponentBegin()

LsTable\_FindRowByContent = LsTable\_FindRowByContent\_Impl( Uri, ColumnTitle,

Operator, Content, Options )

EventWebComponentEnd()

End Function

Validating input parameters: The first statements of the implementation validate the input parameters, to avoid common issues.

Code Syntax

Public Function LsTable\_FindRowByContent\_Impl ( Uri, ColumnTitle, Operator,

Content, Options )

LsTable\_FindRowByContent\_Impl = "

If IsNull(Uri) Or IsNull(ColumnTitle) Or IsNull(Content) Then

Exit Function

End If

If IsNull(Operator) Then

Operator = "="

End If

If IsNull(Options) Then

Options = ""

End If

...

End If

In this example, the URI and some parameters are mandatory, some have a default value. The script checks them all and reacts accordingly.

Writing information to the execution report: The following code just provides feedback to the test engineer. The value of each input parameter is written to the execution report.

Code Syntax

ReportDebugLog "LsTable\_FindRowByContent" & \_

vbCrLf & "Uri: " & Uri & \_ vbCrLf &

"ColumnTitle: " & ColumnTitle & \_ vbCrLf

& "Operator: " & Operator & \_ vbCrLf &

"Content: " & Content & vbCrLf &

"Options: " & Option

Creation of the Query Restricted to a Table Container

If the input parameters are consistent, a query is created using the Light Speed Controller, and it’s scope is restricted to the table by setting theParentControlUri property:

Code Syntax

Dim controller, query, filter

Set controller = LsController() Set

query = controller.CreateLsQuery()

query.ParentControlUri = uri

Definition of the Criteria

To find a row by checking the content of one of the cells, we use a single filter with 3 conditions:

Code Syntax

Set filter = query.SetFilter() filter.AddCondition "tag", "=", "TD"

filter.AddCondition "ls.subtype", "=", "SC" filter.AddCondition

"ls.column~" & ColumnTitle, Operator, Content, Options

The first condition specifies the tag used by the Light Speed Framework to generate a cell. The regular <TD> HTML tag is used.

Note

This condition is not mandatory, but filtering out the other tags has a significant impact on performance, since the number of UI elements matching this condition might be small.

The second condition relies on Light Speed data to determine the sub-type of control. The Light Speed Framework marks all cells with the SC sub-type, regardless of the actual type (such as input field or checkbox). For more information on data, specific to the Light Speed Framework, see [Technical Information for the Query API](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0b6f4d347093480cb24d5734d2691782.html).

The third condition is the most important one; it uses the ColumnTitle parameter specified to search for the cell displayed in the column we are interested in. This condition also uses the Operator, Content, and Options parameters to specify the value we expect in the cells. For more information on boolean operators and options for conditions, see [Technical Information for the Query API](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0b6f4d347093480cb24d5734d2691782.html).

Resolving the Query

The criteria have been defined; the query is now ready to be executed. The following code triggers the query resolution:

Code Syntax

Set control = query.SelectSingle()

If control Is Nothing Then ReportLog "FAILED", "LsTable\_FindRowByContent",

"Operation Failed - Row could not be found" Else

LsTable\_FindRowByContent\_Impl = control.GetRelevantControl().GetLsRow()

...

End If

By calling up the SelectSingle method, the CBTA\_LS\_T\_FindRow component only searches for a single row, that is, the first row that matches the criteria. It is also possible to retrieve a set of cells that match the criteria, by calling up the Select method instead.

Returning the Row Number

The following code gets the row number by using Light Speed Data that is made available by interfaces that are specific to the underlying UI technology.

For more details on Light Speed-relevant control interfaces, see [Technical Information for the Query API](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0b6f4d347093480cb24d5734d2691782.html).

Code Syntax

LsTable\_FindRowByContent\_Impl = control.GetRelevantControl().GetLsRow()

...

Note

The CBTA\_LS\_T\_FindRow component returns the row number in its output parameter. The internal method used to set the output parameter is not in scope of this document.

Selecting the Row

The CBTA\_LS\_T\_FindRow component can select the row as well as search for it. The script has to ask for the URI of the cell that matches the criteria, and then trigger the row selection.

The URI is retrieved by calling the following GetControlUri method, and the selection is made by calling theWEB\_WebControl\_Table\_SelectRow function, which corresponds to the CBTA\_WEB\_SelectRow default component. In the following example, the selection is only made when the Options parameter contains /Select:

Code Syntax

If InStr(Options, "/Select" ) Then

ReportLog "INFO", "LsTable\_FindRowByContent", \_

"Operation succeeded - Row has been selected" & vbCRLf & \_

"Row is: " & LsTable\_FindRowByContent ReportDebugLog

"LsTable\_FindRowByContent - Row Selector Uri: " &

control.GetControlUri() WEB\_WebControl\_Table\_SelectRow

control.GetControlUri()

Else

ReportLog "INFO", "LsTable\_FindRowByContent",

"Operation succeeded - Row is: " & LsTable\_FindRowByContent

End If

* Technical Information for the Query API

This chapter lists technical information, in addition to the preceding documents about the query API.

**Attribute Names and Property Names**

**HTML Attributes and Properties**

You can use the condition to check the regular HTML attributes and properties. The following attribute names are supported:

* id: ID of the HTML element
* tag: tag of the HTML element
* innerText (\*): value of the innerText property of the HTML element
* value (\*): value attribute of the HTML element
* readOnly (\*): value of the readOnly attribute as a Boolean
* disabled (\*): Boolean value of the disabled attribute as a Boolean
* class: list of styles applied to the HTML element
* className: same as class
* href: href attribute of a link, that is, of the anchor HTML element

Caution

(\*) Some HTML attributes may not reflect the visual aspect of the Web controls, depending on the UI technology. For instance, the UI technologies do not necessarily rely on a disabled HTML attribute to enable or disable a control, so the disabled attribute is meaningless. In such a situation, another attribute that is specific to the underlying UI technology can provide the information.

**WebCUIF Attributes**

The WebCUIF attributes that can be used to test CRM applications that start with the prefix crm.. The following attributes are supported:

* crm.value: value of the CRM Web control

The underlying HTML attribute used to retrieve the information can vary, depending on the UI element type. For instance, the value HTML attribute is returned for input HTML elements, but the innerText HTML attribute is used instead for texts and labels.

* crm.id: CRM ID of the CRM Web control

This ID is unique within the HTML content. It has a CRM-specific syntax, and provides the following information:

* + A\_ fragment: specifies the application
  + V\_ fragment: specifies the view name
  + T\_ fragment: specifies the type
  + C\_ fragment: specifies the context node
  + I\_ fragment: is the interface (optional); for input fields, the technical name of the field
  + R\_ fragment: provides the row number (optional)
  + K\_ fragment: provides the key (optional)
* crm.type: type of the CRM Web control

This type corresponds to the T\_ fragment of crm.id. It is independent of the internal HTML tag used to render the UI element. For example, the inputField type is used for checkboxes.

* crm.tagType: tag type or subtype depending on the UI element type.

CRM Web controls of type inputField can have the following tagType settings:

* + checkbox for checkbox controls
  + dropdownListBox for combo box controls
  + valueHelp for input fields with F4 help
* crm.application: A\_ fragment of the crm.id
* crm.view: V\_ fragment of the crm.id
* crm.context: C\_ fragment of the crm.id
* crm.interface: I\_ fragment of the crm.id
* crm.row: R\_ fragment of the crm.id
* crm.key: K\_ fragment of the crm.id
* crm.column.title: title of the column that contains the CRM Web control, if there is an R\_ fragment
* crm.wrongName: returns [wrongName] to indicate an unexpected attribute name that is passed as input parameter
* crm.column:<field>: value of the cell that contains the field specified by the <field> fragment

The <field> fragment provides the technical name of a field that is displayed by the cell. It must be in the format <context>.<interface>.

* crm.column~<title>: value of the cell that is displayed by the column with <title> in its column header

**Light Speed Attributes**

Applications based on the URI Light Speed framework generate controls that include some meta- information, which can be used by queries to search for controls:

* ls.type: type of the control
* ls.subtype: the subtype of the control (if any)

ls.id: ID of the HTML element for the best current control

ls.value: value of the HTML element for the best current control

ls.value: tag of the HTML element for the best control

**Web Dynpro ABAP Attributes**

The following attributes are available only when you test applications that were developed in Web Dynpro ABAP. They provide further information about the fields:

* ls.component: name of the component to which the control belongs
* ls.application: (alias) same as ls.component
* ls.view: technical name of the view to which the control belongs
* ls.field: technical name of the field in the corresponding view
* ls.row: row number (if the control is embedded in a table)
* ls.column.title: title of the column that contains the control if the control is embedded in a table
* ls.column:<field>: value of the cell that contains the field specified by the <field> fragment.

The <field> fragment provides the technical name of the field that is displayed by the cell.

* ls.column~<title>: value of the cell that is displayed by the column with <title> as its column header

**Light Speed Data**

Light Speed Controls have an HTML attribute that provides additional meta information: the ls.data attribute. This information is parsed and made available to the test framework as Light Speed attributes that can be used to define conditions. Not all Light Speed attributes are provided here, because each control can have a different set of data, depending on its nature. You can use the object spy to look for a control, and see the data that is available.

The following are common Light Speed attributes:

* ls.data.value: value of the control
* ls.data.state: state of the control, for example checked for a checkbox
* ls.data.enabled: defines whether the control is enabled (true or false)
* ls.data.readonly: defines whether the control value can be modified (true or false)
* ls.data.text: text of the control

This is usually similar to the innerText HTML attribute, but when the control is complex, the innerText attribute may return a multi-line value (including carriage return characters).

ls.data.text: tooltip associated with the control

**Boolean Operators**

The Query API supports the following operators when you define conditions.

* =: equal to
* ==: same as =
* <>: not equal to
* !=: same as <>
* <: less than
* <=: less than or equal to
* >: greater than
* >=: greater than or equal to
* {contains}: contains
* {startsWith}: starts with ● {endsWith}: ends with
* {matches}: value matches a regular expression; regular expressions use the C# syntax.

**Options for Conditions**

For most of the attributes, the conditions compare operand values as strings, and the comparison is case-sensitive, as in the following example:

filter.AddCondition "crm.column~Partner Function", "=", "Sales employee"

The script may need a case-insensitive comparison. In that case, an additional parameter specifies condition options. Example: filter.AddCondition "crm.column~Partner Function", "=", "SaLeS EmPlOYee", "/u"

With the /u option, both values are converted to upper-case before they are compared.

These are the available options:

* /u: upper-case. Non-case-sensitive comparison. Both operand values are converted to upper-case before they are compared.
* /t: trimmed. Both operand values are trimmed, to remove heading and trailing spaces, before they are compared.
* /b: Boolean. Both operand values are converted to a Boolean value (true or false) before they are compared. This option is implicit for boolean attributes such as readOnly and disabled
* /i: integer. Both operand values are converted to an integer before they are compared. The value is not truncated; for example, the value 10.50 is converted to 11.
* /f: float. Both operand values are converted to float before they are compared.

**Web Query API**

**Web Controller Interface**

The Web controller interface is the main entry point to test Web applications. It exposes the following methods:

* CreateQuery: creates a query object to search the HTML content for Web controls
* CreateWebQuery: same as CreateQuery method of the Web controller
* GetWebControlByUri: resolves the URI parameter and returns the corresponding Web control (if any)
* GetElementByUri: resolves the URI parameter and returns the corresponding HTML element (if any)

**Web Query Interface**

The Web query interface is the main interface of all query objects. It exposes the following methods:

* SetFilter: removes all filters and defines the first one

Input parameters: none. Use the SetCondition and AddCondition methods to define the criteria.

* AddFilter: adds an additional filter

Input parameters: none. Use the SetCondition and AddCondition methods to define the criteria.

* SelectSingle: resolves the query and returns the first Web control that matches the criteria
* Select: resolves the query and returns a set of Web controls

Input parameters: maxHits (optional), specifies the maximum number of controls to return.

* GetMainControl: resolves the ParentControlUri (if specified) and returns the corresponding Web control.

It also exposes the following properties:

* + ParentControlUri: defines the scope of the search for controls

Its value is the URI of the parent container that is to be searched. When it is empty, the query searches the main document of the main page. See also **Reducing the Scope of a Query** and **Searching for Web Controls in a Popup Window**.

**Web Filter Interface**

A filter object that contains a list of conditions that specify the criteria that the controls have to match, and that can have sub-filters that specify additional criteria.

* SetCondition: removes all conditions (if any) and defines the first one.

Input parameters:

* + AttributeName: attribute name (or the property name)
  + BooleanOperator: operator to use to evaluate the condition
  + Value: expected value
  + Options: conversion options

See also **Attribute Names and Property Names**, **Boolean Operators**, and **Options for Conditions**.

* AddCondition: adds a condition

Input parameters: same as for SetCondition.

* RemoveAllConditions: removes all conditions (if any)
* SetFilter: removes all sub-filters and defines the first one

Input parameters: none. Use the SetCondition and AddCondition methods to define the criteria.

See also **Combining Filters for Complex Queries**.

* AddFilter: adds an additional sub-filter

Input parameters: none. Use the SetCondition and AddCondition methods to define the criteria.

* RemoveAllFilters: removes all filters (if any)

**Web Condition Interface**

Conditions are defined using the SetCondition and the AddCondition methods of the Web filter interface. The condition object that is created exposes the following properties:

* AttributeName: name of the attribute or property that is used by this condition
* BooleanOperator: operator that is used to evaluate this condition
* Value: expected value of the attribute or property
* Option: options that are used to evaluate this condition

**Web Control Interface**

The Web control interface provides access to the HTML attributes and properties of the underlying HTML element. It exposes the following methods:

* GetHtmlElement: returns the underlying HTML element

The object returned here implements Microsoft interfaces, depending on its nature.

* GetControlUri: returns the URI of the control in the current HTML content

This URI can be used to call up a default component implementation, to simulate user actions such as a mouse click.

* GetAttribute: returns the value of an HTML attribute of the underlying HTML element Input parameters:
  + AttributeName. Name (value) of the attribute.
* GetProperty: returns the value of an HTML property of the underlying HTML element

Input parameters: PropertyName. Name (value) of the property (for example innerText).

* HighLight: highlights the HTML element at runtime, for troubleshooting

Input parameters: ColorName. The color that is used to highlight the control property (for example innerText), for example red, green, or blue.

* FindFirstChildElement: searches for a child control Input parameters:
  + TagName: HTML tag to search for, for example SPAN
  + Recurse (optional): true to search recursively in the HTML DOM hierarchy of HTML elements
* FindLastChildElement: searches for the latest child control Input parameters:
  + TagName: HTML tag to search for, for example SPAN
  + Recurse (optional): true to search recursively in the HTML DOM hierarchy of HTML elements
* FindChildElementAtPosition: searches for a child control at a position (index) Input parameters:
  + TagName: HTML tag to search for, for example SPAN
  + Index: position number in the set of children
  + Recurse (optional): true to search recursively in the HTML DOM hierarchy of HTML elements

**Web Control Collection Interface**

The set of Web controls is the result of the query that is called by the Select method. The test script can iterate through the set with the following methods and properties:

* ControlAt method: returns the Web control at the specified position Input parameters: Index. The position number in the set of Web controls.

See also **Iterating through a Collection of Web Controls**.

* Count property (read-only): number of controls in the set

**Operation Result Interface**

Interface that provides results when you call a default component implementation. It has the following properties:

* Status (read-only): DONE, PASSED, WARNING, or FAILED
* Feedback (read-only): message that provides feedback about the operation that was performed
* Value (read-only): output value of the operation
* Comment (read-only): additional information (optional)
* ImageTitle (read-only): title of the screenshot (if any)
* ImagePath (read-only): path to the screenshot (if any)
* OutputParameters (read-only): set of output parameters

**Interfaces Specialized for WebCUIF**

**CRM Controller Interface**

The controller interface for CRM exposes the following methods:

* CreateCrmQuery: creates a query object to search the HTML content for CRM controls
* CreateQuery: same as CreateCrmQuery method when using this CRM controller
* CreateWebQuery: same as CreateQuery method of the Web controller
* GetWebControlByUri: resolves the URI parameter and returns the corresponding CRM control (if any)

**Query Interface**

The CRM query interface exposes the same methods as the Web query interface that it extends, but it returns the WebCUIF controls instead of regular Web controls.

**CRM Control Interface**

The CRM control interface extends the Web control interface. It inherits capabilities from the Web control interface. It exposes the following, additional methods:

* GetTestModeControl: returns the control that provides meta- information about the CRM control

This control is usually a hidden SPAN element that the WebCUIF framework inserts in the HTML content when CRM applications run in test mode.

* GetTestModeElement: returns the SPAN element providing meta- information about the CRM control

**CRM Test Mode Control Interface**

This interface provides access to the meta- information that the WebCUIF framework inserts into the HTML content when CRM applications are running in test mode.

| **Method Name** | **Description** | **Equivalent when Filtering** |
| --- | --- | --- |
| GetCrmId | returns the ID of the control | crm.id |
| GetCrmType | returns the type of the control (T\_ fragment of the ID) | crm.type |
| GetCrmTagType | returns the subtype of the control, as defined by the tagType HTML  attribute, if any | crm.tagType |
| GetApplication | returns the application to which the control belongs (A\_ fragment of the ID) | crm.application |
| GetViewName | returns the view the control belongs to (V\_fragment of the ID) | crm.application |
| GetCrmContext | returns the context the control belongs to (C\_fragment of the ID) | crm.context |
| GetCrmInterface | returns the technical name of the control (I\_fragment of the ID) | crm.context |
| GetCrmRow | returns the row number when embedded in a cell in a table | crm.context |
| GetCrmKey | returns the key of the control when child item of a dropdown list box | crm.key |

**Interfaces Specialized for Light Speed**

**Light Speed Controller Interface**

The controller interface for Light Speed exposes the following methods:

* CreateLsQuery: creates a query object to search the HTML content for Light Speed controls

Input parameters: Options. Activates default filtering on common attributes, to restrict the number of elements found by the query.

* + /all: returns all HTML elements that are child of a Light Speed control. This is the default value.
  + /ls: returns all Light Speed elements, which have a ct or subct attribute.
  + /ct: returns all Light Speed elements with a ct attribute
  + /subct: returns all Light Speed elements with a subct attribute
  + /web: returns all HTML elements, even those that are not children of a Light Speed control Note: The option chosen here impacts the performance of a query.
* CreateQuery: same as the CreateLsQuery method when using this Light Speed Controller
* CreateWebQuery: same as the CreateQuery method of the Web controller
* GetWebControlByUri: resolves the URI parameter and returns the corresponding WebCUIF control, if any

**Light Speed Query Interface**

The Light Speed query interface exposes the same methods as the Web query interface that it extends, but the query returns objects that implement the Light Speed controls instead of Web controls.

**Light Speed Control Interface**

The Light Speed Control interface extends the Web control interface. It inherits capabilities from the Web control interface. It exposes the following, additional methods:

* GetRelevantControl: returns the relevant control, the one with HTML attributes (such as ct or subct), which define the control type
* GetRelevantElement: returns the relevant element, the underlying HTML element of the relevant control, the one with HTML attributes (such as ct or subct), which define the control type

**Light Speed Relevant Control Interface**

The relevant control interface only makes the access to some meta- information easier, that the Light Speed framework provides.

| **Method Name** | **Description** | **Equivalent when Filtering** |
| --- | --- | --- |
| GetLsId | returns the ID of the control | ls.id |
| GetLsTag | returns the tag of the underlying HTML element | ls.tag |
| GetLsType | returns the type of the control | ls.type |
| GetLsSubtype | returns the subtype of the control | ls.subtype |
| GetLsComponent | returns the component of the control (only Web Dynpro ABAP) | ls.component |
| GetLsViewName | returns the view of the control (Web only Dynpro ABAP) | ls.view |
| GetLsFieldName | returns the name of the control (only Web Dynpro ABAP) | ls.field |
| GetLsRow | returns the row number (when embedded in a cell in a table) | ls.row |

# Transport Management for Tests and Components

## Use

The transport of the objects in the Extended Computer Aided Test Tool (eCATT) and of the metadata in the Test Composition Environment (TCE) that defines the CBTA tests and screen components is integrated. This enables CBTA tests to be transported.

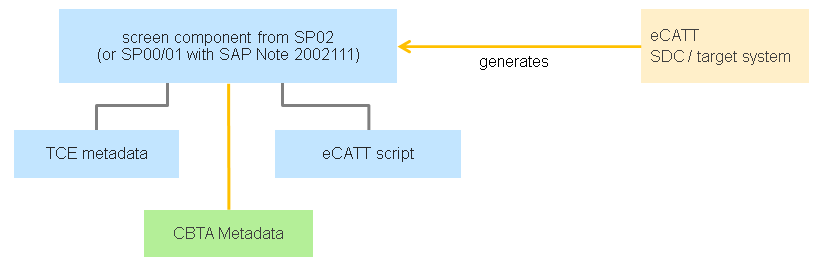
CBTA uses a repository to manage the CBTA screen components. It stores metadata that is used by the component generation process, with which CBTA can organize the components by catalog. A catalog of screen components is valid for an eCATT system data container and target system, to represent a consistent system landscape.

Note

If you use TCE, apply SAP Note [1871671 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1871671)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1871671) (Missing TR number in BAdI call from CL\_APL\_ECATT\_OBJECT). Otherwise, metadata for TCE and CBTA is not in your transport request.

In contrast to transport management, the download and upload of test scripts that have not been enhanced is not supported for TCE and CBTA test scripts.

CBTA generates one screen component name per system data container and target system, at recording time (enablement of Screen component option during the recording needed). The metadata of the CBTA repository determines the component name.



A screen component is generated by an SDC and target system, which define its validity. A screen component is still executable for other SDCs/TSs, because the test execution engine does not check its validity and always tries to execute it because it might still be compatible.

## Features

Automatic Transport Selection

The request of the master test script is used to store a new or updated screen components. (This is the one you choose from the Test Composition Environment when you save your master test.

Note

A CTS-related message popup can appear and remain in the background. It blocks the process flow in the CBTA wizard. Close it with OK.

Organization of Transport Requests and Tasks

If a screen component that is shared between business processes and related tests is already part of another transport request, it will remain in the original request without any notice. This is default CTS object handling. No consistency check is made for the request when you release it. It is possible that you transport a CBTA test with only a subset of the CBTA screen components that are involved in the scenario. In this case, when you open a test in your transport target system, the test workbench will raise errors about the missing components until the test and its usage of components is consistent. Once the missing screen components are transported, you can execute CBTA, if you defined the system data container, target component, and tester profile, separately.

To avoid this problem, organize transport requests and tasks based on the scope of your project. Because screen components are dedicated to one system data container and target component, you can create one transport request to be used by all test engineers who are involved in the related test landscape. If the test scope is too large and a synchronized task release too difficult, you can create one transport request for each functional area, as long as cross-business components are properly managed.

FAQ

Can I force the transport of a screen component?

A CBTA screen component is just an eCATT script that is enriched with TCE and CBTA metadata, so you can use any existing standard entry points for eCATT objects transport to transport a screen component, manually.

Are the system data container and target component (assigned to the CBTA master test) also transported?

When you save a CBTA test, the system data container and target component of the test are not automatically added to the transport request. You can define this in transaction SECATT. Enhanced SDC data is not transportable, and must be redefined in the target system with the SUT management tool. This is not really relevant, because the underlying RFC destination and user credential data cannot be transported, anyway.

The SUT management tool can be accessed from the Administration - Test Suite Test Automation Framework  System Under Test Management.

**Configuration of the Test Suite**

## **Use**

Basic parts of the setup take place in **SAP Solution Manager Configuration**. It is completed by several other activities that are carried out on demand.

## Activities

1. Start the **SAP Solution Manager Configuration** (transaction SOLMAN\_SETUP) and follow the guided procedures under **Test Suite**.

This is available for Test suite preparation, test automation preparation, component-based test automation, and business process change analyzer.

For more information, see the help texts for the steps in the guided procedure.

1. In the **Administration - Test Suite** application, you can make further basic settings. For more information, see [Administration](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/cf60aa80c5c84698bc794168c392b5f2.html).
2. The complete list of settings for the test suite is in the customizing of SAP Solution Manager (transaction SPRO), under **Capabilities**  **Test Suite**  **Test Suite for SAP Solution Manager**.

For more information, see the documentation for each customizing activity.

**Setting Up the Test Tool**

Before you can create test configurations for automated test cases, you need to set up the required test tools.

1. In the **Administration - Test Suite** application, choose **Test Automation Framework**  **Test Tool Attributes Definition**.
2. You can check and edit the attributes which a test tool can supply to SAP for a test script. For partner test tools, refer to the documentation of the tool vendor.
3. Save and close the window.

* the **Central Test Workbench Settings** application help in SAP Help Portal at [http://help.sap.com[Information published on SAP site](http://help.sap.com/disclaimer?site=http://help.sap.com)](http://help.sap.com/disclaimer?site=http://help.sap.com).
* [Assigning Defect Types to Error Types](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/51deab516c972166e10000000a441470.html)

**Registering a Client-Side Automated Test Tool or a Third-Party Test Tool**

You can register a client–side automated test tool (CBTA or a third-party test tool) with which you want to create automatic test cases.

**Prerequisites**

* You have read SAP Note [519858 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/519858)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/519858).
* You have installed a certified external tool on the local machine of the user or the tool is available via a Citrix client. You register the tool in the customizing (transaction SPRO), under **SAP Solution Manager**  **Capabilities**  **Test Suite**  **Test Suite for SAP Solution Manager**  **Test Automation Framework**  **Setup**  **Register Test Tool**:
* You have generated a standard user in the customizing (transaction SPRO), under **SAP Solution Manager**  **Capabilities**  **Test Suite**  **Test Suite for SAP Solution Manager**  **Test Automation Framework**  **Setup**  **Generate Users**.
* The third party tool must use the predefined user ECATT\_ET\_USR for the communication with SAP Solution Manager. The roleSAP\_ECET is assigned to this user.
* An SAP GUI 7.40 is installed on the front-end computer.
* GUI scripting is activated on the front-end computer.

**Procedure**

1. In the customizing of SAP Solution Manager (transaction SPRO), go to **Capabilities**  **Test Suite**  **Test Suite for SAP Solution Manager**  **Test Automation Framework**  **Setup**  **Register Test Tool**.

You go to the **Customizing Table for External Test Tools** screen.

1. Choose **Change**.
2. Choose **New Entries**.
3. Under **Tool Name**, select a test tool and enter the required information.

For more information about how to enter the required information, see the user documentation of the respective third-party tool.

1. Choose **Continue**.
2. Save and close the window.

# Administration - Test Suite

In the Administration - Test Suite application, you can perform further customizing for the test suite, in addition to the setup activities in SAP Solution Manager Configuration and in the customizing (transaction SPRO).

* Preparation tab:
  + Test Case Status: Customize the statuses a tester can assign to a test case during test execution.

For more information, see the instructions in the customizing (transaction SPRO) for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Preparation  Administration  Define Test Status Values.

* + Note and Test Result Template for the test documentation (notes and test results): Upload a custom Microsoft Word template, to allow testers to create a note or a test result using the template when executing a manual test case.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite with SAP Solution Manager  Preparation  Administration  Test Documentation (Notes)  Define Template for Test Documentation (Notes) and Define Template for Test Documentation (Notes).

* + E-Mail Notification Settings: Assign existing smart forms for e-mails to notification types. These e-mails are sent to testers for test execution and to inform about test errors.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Preparation  Setup  Standard Configuration  E-Mail Notifications  Assign E-Mail Forms for Notifications.

In the same customizing branch, you can find the following options:

* + - Register Users for Receiving E-Mail Notifications
    - Define E-Mail Forms for the Test Suite
  + Maintain Exclusion List for Test Plan Filters: Define which attributes are to be hidden from the lists in test plan creation, to simplify the activity for users.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Preparation  Administration  Maintain Exclusion List for Test Plan Filters.

* + Automatic Recording of Test Effort: Define the automatic time recording of the test effort for manual test cases.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Preparation  Administration  Activate Automatic Recording of Test Effort for Manual Test Cases.

* + Default Required Execution Time of Test Case Types: Define the default execution times that are expected for the different types of test cases.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Preparation  Setup  Standard Configuration  Central Test Suite Settings  Specify Default Test Case Execution Times, Depending on Type.

* + Definition of Test Classification: Define test categories, for example, unit test, functional test, integration test, user acceptance test, regression test. For more information, see the instructions in the customizing for SAP Solution Manager, under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Preparation  Administration  Define Test Classification.
  + Definition of Template for Test Result Report: SAP delivers a template that is used to configure the test reports. You can define your own template if you want to change the cover and end page, documents keywords, and other properties.
* Test Automation Framework tab:
  + Test Tool Attributes Definition: Register test tool-specific attributes to be transferred to the tool during test script recording.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Test Automation Framework  Administration  Register Test Tool Attributes.

For more information, see [Maintaining Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html).

* + Maintain SUT Systems: Define the connection and user credentials for systems under test in a test profile, so you do not have to define a separate RFC destination for each business user of your test.

For more information, see the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Component-Based Test Automation  Administration  Configure System Under Test.

For more information, see [SUT Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/69c71656795f5f5ae10000000a4450e5.html).

* CBTA tab:
  + Maintain Client Settings: Configure the CBTA (component-based test automation) client that you have installed locally.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Component-Based Test Automation  Administration  Configure CBTA Client.

* + Customize RTL: Customize the runtime library for CBTA. You can add custom functions, for example recording or playing tests with Adobe Flash technology or Java applets.

For more information, see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suite for SAP Solution Manager  Component-Based Test Automation  Administration  Customize Runtime Library.

* + Perform Self-Checks for the CBTA tool and systems under test. Verify the correct installation and configuration of the CBTA client.

For more information see the instructions in the customizing for SAP Solution Manager under Capabilities  Test Suite  Test Suiter for SAP Solution Manager  Component-Based Test Automation  Administration  Perform Self Checks.

# Test Repository: Test Scripts and Configurations

## Use

The applications **Test Repository - Test Configurations** and **Test Repository - Test Scripts** give you access to the test composition environment in SAP Solution Manager. Here, you can create and edit test scripts and test configurations for automated test cases:

* A **test configuration** contains a system data container, a test data container, and one test script. The data from these elements can be reused in several test configurations.
* A **test script** describes in technical detail the components and sequence of activities that are to be tested, for a selected business process. The test script can contain one or several steps, which you can put into the right order.

You can create test scripts automatically by using test tools that are configured in the system, either eCATT, component-based test automation (CBTA), or third-party.

You can reuse existing test scripts, create modular test scripts and test end-to-end processes. For more information, see[Creating a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3aaf17528d699a33e10000000a44538d.html).

* A **test case** is the assignment of a test configuration to a structure node (for example a business process step) in your solution documentation. For more information, see [Solution Documentation: Creating Test Cases](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/df49e0555937e263e10000000a44538d.html).

To get the proper SAP Solution Manager context, it is recommended to create the test configuration from Solution Documentation.

## Prerequisites

* To use a third party test tool, you have installed the test tool on the client machine where SAP GUI is installed for the SAP Solution Manager system and for the systems under test, and you have registered it in SAP Solution Manager.
* You have defined your solution documentation and created the relevant executables there. Selection of an executable during creation of automated test is a mandatory step, because this way the system under test will be derived.
* Optional: You have created a test data container. For more information, see [Creating Test Data Containers in the Test Automation Framework](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/90c365e3a84d4c5c9dae538b3d2ae17e.html).

## Activities

Perform the following activities:

1. Create or edit a test configuration containing a test data container, a system data container. For more information, see [Creating Test Configurations and Scripts](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/fb9bf93de5534a43862558adf678e5bc.html).

A bare test script which does not yet contain test script steps, is created automatically. You create a test script only, without test configuration, for example to create a test quickly.

1. To specify, for example, the application component or responsible persons, maintain the SAP attributes of the test configuration or test script. For more information, see [Maintaining Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html).
2. To edit the test script, record a test script or define the test script steps manually. For more information, see [Creating a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3aaf17528d699a33e10000000a44538d.html).

You can compose an end-to-end test configuration of steps which you have created in various test tools: eCATT, CBTA, standard regression testing or third-party test tools.

1. Execute the test script manually, and resolve errors. For more information, see [Executing Test Configurations or Test Scripts](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/15ba145224809c60e10000000a44176d.html).

To monitor a test execution in detail, you can display an execution report. For more information, see [Test Execution Reports](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0a71135296bcb964e10000000a423f68.html).

1. Use the solution documentation in the **Test Suite - Test Preparation** application to create a test case for a business process step. While creating the test case, you assign a test configuration to it, or create a new one. For more information, see [Solution Documentation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3ef70f5452e7024be10000000a44176d.html).
2. To schedule the test case for execution, assign it to a test package in a test plan. For more information, see [Test Plan Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/48e151fddb163184e10000000a421937.html).

# Creating Test Configurations and Scripts

## Use

In the **Test Repository** applications, you create and edit test configurations and test scripts. A test script is part of each test configuration and describes the technical components that are required to execute the business process step. For more information, see [Components of Automated Test Cases](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4a6b8db0db364e17ae8e9e6c4451c363.html).

You assign a test configuration and test script to a solution, which is linked to a system data container. You can assign a test data container to a test configuration.

You create a test configuration to create an automated test case. Since there cannot be a test configuration without a test script, you have to create the related test script in parallel. Later, you can edit an assigned test script or replace it with a different test script.

You can automatically create the test script by recording a sequence of steps related to an executable (for example, a transaction or Web Dynpro application) with the CBTA tool, for example. You can also us eCATT or third-party test scripts.

## Prerequisites

See the prerequisites under [Test Repository: Test Scripts and Configurations](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b5e6e0ffd8474be08453a43b310f7cc3.html).

Caution

Before you edit an existing test configuration, check in which test plans, test packages, or test scripts it is used. To do so, select the test configuration and choose **Where-Used**.

## Procedure

**Creating Test Configurations in Solution Documentation (recommended)**

We recommend that you create a test configuration from the solution documentation in the **Test Suite - Test Preparation** application, for the following reasons:

* In the **Test Suite - Test Preparation** application, you can select a specific process step in the right solution and branch, first, and directly create the test case with the test configuration to it. So the correct assignment exists from the beginning.
* You can select the executable object for the process step, in the solution documentation, or directly get the executable that is assigned to it.
* You can assign several test configurations to a process step and organize your tests accordingly.
* You can use the business process change analyzer (BPCA) for the solution.

After creation, you can also edit test configurations in the **Test Repository - Test Configurations** application.

**Create a test configuration as follows:**

In the **Test Suite - Test Preparation** application, open a solution and branch.

In the **Browser** (selectable at the top of the new window), drill down to the business process step for which you want to create a test.

In the table of elements, you see the executables and test cases that have been assigned already.

To assign a new executable to the business process step, right-click in this table and choose **New**  **Executables**. Such an executable can be a transaction, program, SAP GUI transaction, a CRM application (using the role and logical link parameters), a Web Dynpro ABAP application, or a URL.

As a consequence, the executable is available to be assigned to a test case, in the next steps.

To create and assign a new test case to the business process step, right-click in the table with elements and choose **New**  **Test Cases**  **Test Configuration (Create)**:

1. The editor for the test configuration opens.

Fill the mandatory fields and save.

1. Complete the settings on the **Attributes** tab.

For more information, see [Maintaining Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html).

1. Assign the executable to be tested, as the system where the test will run is derived from the executable.
2. If you chose the CBTA (component-based test configuration) as test tool, the assignment of the executable will also be used during the recording of your test. After selecting the executable, you can choose **Launch CBTA** to directly start the creation of the test script by running the selected executable in CBTA.
3. When the test configuration is complete, choose **Save** and **Close**.

Back in the editor for the business processes, select the newly created test case.

On the right-hand side, you can define further characteristics:

* **Test Classification**: You can define the test case as regression test or user acceptance test, for example. This is helpful when you create test plans, later, for these specific purposes.
* **Test Data Variant**: You can select it for automatic test cases. The test data contains a set of values to be used for the automatic test. The variant defines which specific sets are to be used,
* **Target Logical Component Group**: When you create a test configuration, you have to define the logical component group. Together with the branch and system role, this will allows to execute the test in the correct target system. The logical component group specified here reflects the one that is selected for the test configuration.

Save and close the window.

You can edit test configurations in the **Test Repository - Test Configurations** application or in the solution documentation in the **Test Suite - Test Preparation** application.

**Creating Test Scripts in the Test Repository (for simple tests)**

We recommend that you create test configurations including the test script in the **Test Suite - Test Preparation** application. But to create a very simple test, quickly, you can create the bare test script directly in the **Test Repository - Test Scripts** or **Test Repository - Test Configurations** application:

1. Start the **Test Repository - Test Scripts** or **Test Repository - Test Configurations** application. When you create or edit a test configuration, the test script is automatically included.
2. Choose **Create**.
3. Select a solution.
4. You can define the system role directly in the test repository. Together with the solution and branch, it allows to automatically derive the physical system, to which the logical component group points. Changing the system role changes the system on which the tests are recorded and executed. This is useful when you define a set of landscapes which contain a set of systems like development, test, pre-production, and production systems.
5. If you create a test configuration: In the **Test Configuration** field, enter the name of the test configuration.

Note

If you want a test configuration to remain local (not be transported to other systems) enter a technical name starting with Z(depending on the name space specified in your system configuration). For example, you store all your test cases on a single SAP Solution Manager system. The name space is automatically assigned after recording a screen component. For more information about configuring the name space, see SAP Note [1763697 [Information published on SAP site](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1763697)](http://help.sap.com/disclaimer?site=https://launchpad.support.sap.com/#/notes/1763697).

1. In the **Test Tool**field, select the test tool.
2. Enter the name of the test script to be created, or assign an existing test script, if you create a test configuration.

Note

If you do not assign an existing test script, a test script is created automatically when you create a test configuration. The name of the test script is the test configuration name.

1. Optional: Change the default version number.
2. In the **Title** field, enter a name.
3. In the **Package** field, select a transport package or **Local Object**.
4. Confirm.

# Defining Test Configuration Attributes

## Prerequisites

* You have the authorization of a development consultant.
* If you use the tool for component-based test automation (CBTA), you have created a test profile. For more information, see[Creating a Test Profile With Logon Data for SUTs](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/ca12125207659933e10000000a44538d.html).

## Context

You can edit a test configuration that you have created as described under [Creating Test Configurations and Test Scripts](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/fb9bf93de5534a43862558adf678e5bc.html).

## Procedure

1. In the **Test Repository - Test Configuration** application, click on a test configuration to edit it.
2. On the **Attributes** main tab, on the **Test Configuration Attributes** sub-tab, the most important attributes are displayed.

(**Note**: If you have created the test script with a third party test tool, there is also the **Test Tool Attributes** sub-tab. It is read-only and the attributes are filled during the recording.)

First, enter the test configuration attributes, and define the test script attributes afterwards.

1. Assign Executable: Assign the executable that is to be tested. For more information, see [Executable Types](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/f974f851579b7d38e10000000a445394.html).

When you assign the executable from the related solution, the system automatically assigns the logical component group.

1. The application component is derived from the selected executable. If this is not available, you can enter it manually. It is usually the same as that of the tested applications.
2. Enter the following data to describe the test script or test configuration.
   * **Person Responsible**: By default, the current user is predefined. You can change it.
   * Assign a test profile that was defined in the management for systems under test. For more information, see [Creating Test Profiles and Logon Data for SUTs](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/ca12125207659933e10000000a44538d.html).
   * To test the logon to the target system click the link to the target system.
   * **Status**: The status is set to active by default. Only active scripts are included in automatic tests. If you do not want the test script to be executed (because it is not complete, or you are modifying it, for example), set the status to inactive.
   * **Time Required**: Enter the time that the test configuration is supposed to take.
   * **Priority**

# Assigning Executables to Test Scripts

## Use

Executables are created in the [Solution Documentation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3ef70f5452e7024be10000000a44176d.html). So, when you assign a test case to a solution, you can assign the executable of the solution to the test case.

This assignment is mandatory because it allows the determination of the target system. The system will offer a search help to retrieve the available executables according to your solution documentation context.

## Activities

1. In the **Test Repository - Test Scripts** view, select a test script for editing.
2. On the **Attributes** tab, assign an executable.

Define the logical component group, executable type, and executable and choose **Search**.

The search results are retrieved from the related branch of the solution.

1. Click on a search result to assign it to the test script.
2. Save your entries and close the window.

# Creating a Test Script

## Use

The test script is the central part of a test configuration. It describes in detail the components of the business process that are to be tested, and in which sequence. In the **Test Repository** views, you can create and edit test configurations and the included test scripts. For the creation of test scripts, you can use a test recorder to create test scripts automatically and to edit them manually. We recommend that you use the component-based test automation (CBTA) tool because it allows automatic recording and manual editing, which is not supported by many other test tools.

To create your test configuration and test script with the correct SAP Solution Manager context, we recommend that you create the test configuration from the solution documentation in the **Test Suite - Test Preparation** application of SAP Solution Manager.

## Prerequisites

You have completed the test configuration attributes as described in [Defining Test Configuration Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html).

## Procedure

1. In the **Test Suite - Test Preparation** application, open a solution and branch.
2. Navigate to the business process or step for which you want to create the test.

Right-click in the list of elements at the bottom and choose **New**  **Test Cases** and create a new test configuration or assign an existing one.

1. When the test configuration attributes are complete, choose the **Launch <test tool>**.
2. The recording tool starts.
   * If you are using the CBTA test tool, the test creation wizard starts.

For more information, see [Recording a Test Script Using Component-Based Test Automation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/1694d2afbff24a4c80f5eb145c0ef212.html).

* + If you are using the eCATT the test tool, you go to the eCATT Test Script Editor screen.

For more information, see the **eCATT: extended Computer Aided Test Tool** application help in SAP Help Portal at [http://help.sap.com[Information published on SAP site](http://help.sap.com/disclaimer?site=http://help.sap.com)](http://help.sap.com/disclaimer?site=http://help.sap.com), under **Developing Test Scripts**  **Creating Test Scripts**  **Test Script Editor**.

* + If you are using a third-party test tool, the test script creation process starts.

**Combining Test Script Steps from Different Tools**

You can compose an end-to-end test configuration of test script steps that you have created in different test tools: component-based test automation (CBTA), eCATT, or third-party test tools.

You typically define the sequence of the test script steps to be executed. Then you define the parameter values for the steps. For more information, see [Editing a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/098b949b6f6543ac990506e08d85424f.html).

To create a composite test script, create a test configuration or a test script as described in [Creating Test Configurations and Test Scripts](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/fb9bf93de5534a43862558adf678e5bc.html).

In the **Test Tool** field, select **Composite Test**. For more information, see [Defining Test Configuration Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html).

# Editing a Test Script

## Prerequisites

You have created a test configuration including a test script and maintained the test configuration attributes as described in [Defining Test Configuration Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html) and [Creating a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3aaf17528d699a33e10000000a44538d.html).

## Context

When you have recorded a test script, you can edit and complete it:

* You can edit components of a test script that was recorded with the Component-Based Test Automation (CBTA) tool in the **Test Repository** views. For more information, see [Editing a CBTA Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/cb885c577dbd0950e10000000a441470.html).
* You can edit steps (modules) from an eCATT test script or from a eCATT test script combined with third-party test tool steps. For more information, see **Combining Test Script Steps from Different Tools** in[Creating a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3aaf17528d699a33e10000000a44538d.html).
* **Restrictions:**
  + If a test script was modified in eCATT, you can no longer edit it in the **Test Repository** views. To edit the test case, open the test configuration and choose **Start eCATT**.
  + If you use a third-party test tool, the **Test Script** tab page is not displayed in the editor for test configurations. The content of the test script can only be interpreted by the third-party tool, not by SAP Solution Manager.

## Procedure

1. In the **Test Repository - Test Configuration** view, select the test configuration that includes the test script you want to edit.
2. Choose the **Test Script** tab.

The test script steps are displayed. In an eCATT or composite test script, the steps can be test scripts of any test tool type.

1. You can add test script steps, under **Test Script Steps**:
   * Enter the following data for an eCATT or composite test script:
     1. In the **Type** column, select the type of the test tool with which the test script was created.
     2. In the **Name** column, select the test script.
     3. In the **Version** column, enter the relevant version, which is 1, in most cases.

Save your entries. Empty lines are removed automatically.

1. To define the sequence of test script steps, use the **Up** and **Down** buttons.

To duplicate several steps with a test script, you can use the copy and paste feature.

To remove a superfluous step (for example, because you have performed it twice, by mistake) choose **Remove**.

1. Configure parameters under **Test Script**  **Parameters**:
   * To define the values passed to the test script steps (import and export parameters), edit the **Parameters** sub tab page:
     1. In the **Usage** field, define the usage type:
        + No value: No parameter value is passed to the parameter of the test script. The test script parameter does not have a default value.
        + **Fixed**: A constant value is passed to the parameter. The constant value does not depend on the execution of the script.

Example

If you always want to create a sales order for 5 pumps, enter 5 for the parameter **Quantity**, in the **Value**column. If you set usage to **Fixed** but do not enter a value, the value NULL is used in the test script. To define an empty string value, use the value %BLANK%.

* + - * **Exposed**: The parameter is mapped to the parameter of the parent script. The parameter value is passed from (import parameters) or passed to (export parameters) the parameter of the parent script.

Import parameters can receive the values from the test data container.

Several exposed import parameters can receive their values from the same parent script parameter.

A single exposed export parameter can pass its value to several export parameters of the parent script.

Example

You want to create a main test script comprising 3 steps (scripts) with the following parameters:

* + - * + Create quotation

Import parameters: Customer, Material, Quantity

Export parameter: Quotation Number.

* + - * + Create sales order from quotation

Import parameter: Quotation Number

Export parameter: Sales Order Number

* + - * + Create delivery from sales order

Import parameter: Sales Order Number

Export parameter: Delivery Number

You put 3 of the steps in the sequence. To be able to execute the main script for a different customer, material and quantity, you have to know the delivery number created at the end. To do so, set the usage type to **Exposed** for the following parameters:

* + - * + Import parameters Customer, Material, Quantity of the script “Create quotation”
        + Export parameter Delivery Number of the test script “Create delivery from sales order”
      * **Local**: The value passed to the parameter is mapped into the parameter of another step. The value actually used during the execution of the test script depends on the execution of the parent script and the values created in the preceding steps.

Example

You want to pass the value of parameter Quotation Number created by the “Create Quotation” test script step to the “Create sales order from quotation” test script step. To do so, set the usage type to **Local** and map these 2 parameters between them. To do so, in the **Reference Parameter** column, assign the name of one parameter to the other.

* + 1. Define the other columns as required.
  + To display SAP attributes of an eCATT test script or of a test script created with a third-party test tool, choose the **Attributes** sub tab page under **Test Script**.

To change the attributes, under **Test Script Steps**, in the **Name** column, select the test script.

* + To display the description, if available, choose the **Description** sub tab page.

For test scripts steps created with a third-party test tool, to select either the SAP description or the third-party test tool description, select the respective radio button.

* + To display the screen capture of a test script step, if available, choose the **Image** tab page.
  + To check where a test script is used, select the test script and choose **Where-Used**.
  + To edit or display a test script step in a new session, in the **Name** column, choose the test script.

If the third-party tool in which the script was created is not available, you go to the eCATT wrapper test script.

1. To execute the test, choose **Execute**.

# Defining Input and Output Parameters

## Prerequisites

You have created a test script or a test configuration. For more information, see [Creating Test Configurations and Test Scripts](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/fb9bf93de5534a43862558adf678e5bc.html).

## Context

On the **Parameters** tab page in the editor for test configurations, you can edit the export and import parameters of the test script, to define the test script interface. These test script parameters will appear as parameters for the test configuration to which the test script is assigned. You can subsequently use it to assign a test data container that contains the test parameter values.

## Procedure

1. In the **Test Repository - Test Configuration** view, select a test configuration.
2. Choose the **Parameters** tab.
3. To display the parameters, according to the order of the corresponding test steps in the test script, choose **Parameter Default Sequence**.
4. You can edit the parameters of test script steps recorded with eCATT or component-based test automation (CBTA). You can, for example, assign more meaningful parameter names and descriptions, which make it easier to use the test script.

For more information, see [Editing a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/098b949b6f6543ac990506e08d85424f.html)

1. To navigate to the test script steps and their parameters, choose the component or component parameter.

This takes you to the corresponding entries in the **Test Script** tab.

# Further Settings for Test Configuration and Test Script

## Use

In the **Test Repository** views, you can define further attributes for test configurations or test scripts. You can, for example, assign test data containers for reuse, or test data variants.

## Prerequisites

You have defined the test configuration, as described under [Defining Test Configuration Attributes](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/a87c15e5b85f4147a9eba691912b4546.html).

You have defined the the test script, as described under [Editing a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/098b949b6f6543ac990506e08d85424f.html) and [Defining Input and Output Parameters](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/b769b73c138846dcb7015c43ca5bac26.html)

## Procedure

**Adding Test Data Containers**

To reuse a test configuration, but change the values that the test uses, you can add references to one or more test data containers.

1. On the **Test Data** tab, choose **Add**.
2. In the **Alias** column, enter an alias.

When parameters of the test script are mapped to the parameters of a test data container, the references are displayed. If the name of a test data container is too long, it can be difficult to recognize the references. To monitor the mapping, enter an alphanumeric name of up to three characters for the test data container.

1. In the **Test Data Container** column, select the test data container.

In the **Title** field, the name of the test data container is displayed.

If the test data container contains external test data variants, the corresponding checkbox is selected, and the name of the file containing the external test data variants, is displayed.

1. Save your entries.

The test data container is assigned to your test configuration.

1. **To create test data variants for a test configuration, you can to the following:**
   * You can choose whether to display the value of a parameter or the test data container alias and the number of the variant from which the parameter value is taken.
     + To display the value of the parameter, choose **Value**. 200, for example, is displayed.
     + To display the test data container alias and the parameter name, choose **Reference**:

Example

If you have a test data container with the alias TD1, instead of the value 200 you see <parameter name>(<test data container alias>,<number of the test data container variant>), for example I\_ORDER\_TYPE(TD1,1).

* + **To assign a test data variant to the parameters of your test configuration, do the following:**
    - Choose the **Test Data Assignment Wizard** pushbutton. The pushbutton is only active if the test script has parameters.

The Variant Maintenance Wizard starts.

* + - .Enter data as required.
    - Choose **Transfer Wizard Result to Configuration**.
    - On the **Test Data** tab page, choose **Refresh**.

A dialog box appears.

* + - To confirm, choose **Yes**.

Under **Test Data Variants**, the test data variant is displayed.

* + - To allow the test data variant to be selected for execution, select the **Execute** checkbox.

If you select multiple test data variants, you can specify the one to be executed, in the **Start Options** screen, when executing the test case.

For more information, see the **eCATT: extended Computer Aided Test Tool** application help in SAP Help Portal at [http://help.sap.com[Information published on SAP site](http://help.sap.com/disclaimer?site=http://help.sap.com)](http://help.sap.com/disclaimer?site=http://help.sap.com), under **Executing Test Scripts**  **Start Options**.

* + - Save your entries.

**Defining Search Terms**

To find test scripts and test configurations easily, you can assign search terms. Search terms can describe, for example, the scenario for which the test is to be carried out, the user, or the use case.

1. In the **Test Repository** applications of SAP Solution Manager, select a test configuration or a test script.

Search terms you enter for a test script, also apply to the related test configuration and vice versa.

1. Choose the **Search Terms** tab.
2. Enter search terms according to the conventions of your organization. The maximum is 20 characters.

Search term values are converted to upper case.

1. Save and close the window.

**Replacing a Test Script in a Test Configuration**

To replace a test script that was automatically assigned to a test configuration, you can do the following:

* To assign a copy of an existing test script, which you plan to change, choose **Assign Script**  **Copy Existing**.
* To create a reference to an existing test script, choose **Assign Script**  **Existing**.
* To create a new test script, choose **Assign Script**  **New**.

**Using Expert Settings for eCATT**

Optional: To use advanced functions to edit the test script, choose **Goto**  **Expert mode** (transaction SECATT).

Note

If you edit a test configuration or test script with transaction SECATT, you cannot edit it again later in the test composition environment.

# Managing Parts of Automated Test Cases

## Context

You can edit, copy, rename, or delete the following parts of test configurations:

* Test scripts (see [Editing a Test Script](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/098b949b6f6543ac990506e08d85424f.html))
* Test data containers
* System data containers

In SAP Solution Manager, we recommend that you use the **Test Repository** views to edit test scripts and test configurations. Only if you cannot make changes here, use the Extended Test Automation interface in transaction eCATT.

System data containers (SDCs) are automatically created when you modify the solution landscape. SDCs are only used to allow communication with the eCATT layer. They determine the connection settings using [SUT Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/69c71656795f5f5ae10000000a4450e5.html). It is no longer relevant to update SDCs in the Extended Test Automation.

## Procedure

1. In transaction SECATT, select the test configuration, test script, test data container, or system data container you want to maintain.
2. You can do the following:
   * Choose **Display** or **Edit** to open the editor.

The appropriate editor opens.

* + Choose **Delete Object**.

A dialog box appears.

* + - If you selected a test configuration, you can choose whether to select the test configuration only or the test configuration with the current version or with all versions of test scripts. If you delete the test configuration only, you can reuse the respective test script, when you create a new test configuration.
    - If you selected a test data container or a system data container, which is assigned to a test configuration, you have to remove the assignment first.

To do so, choose **Where-used** , to display and edit the respective objects.

Choose **Yes**.

* + To copy or rename the object, do the following:
    - Choose **Copy Object** or **Rename Object**.

A dialog box appears.

* + - Enter the name of the object to be copied or renamed.
    - To copy or rename the test configuration without copying or renaming the respective test script, deselect the **Including Test Script** field.

If you copy a test case, the test script of the source test case is assigned to target test case.

If you rename a test case, the test script of the source test case is assigned to target test case without being renamed.

* + - Choose **Copy** or **Rename**.

1. Save your entries.
2. Exit the function.

# Executing Test Configurations or Test Scripts

## Use

You can schedule the execution of a test configuration in a test plan. In a test plan, test configurations can be run automatically, or manually, by testers.

You can also execute a test configuration or test script manually, without a schedule, to ensure that the test works correctly.

## Procedure

**Executing a Test Manually**

1. In the **Test Repository** views, select a test script or test configuration.
2. Choose **Execute**.

A new window opens, where you can edit the start options.

For more information on eCATT tests, see the application help **eCATT: extended Computer Aided Test Tool** at [http://help.sap.com[Information published on SAP site](http://help.sap.com/disclaimer?site=http://help.sap.com)](http://help.sap.com/disclaimer?site=http://help.sap.com), under **Executing Test Scripts**  **Start Options**.

1. Choose **Execute**.
2. Check the execution report. For more information, see [Displaying the Execution Report](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/0a71135296bcb964e10000000a423f68.html).

**Scheduling the Execution of Tests in Test Plans and Test Packages**

# Test Cases

## Use

Test cases, which include the test configurations, are assigned to business processes in the solution documentation. This is done in the **Test Suite - Test Preparation** application where you can do the following:

* Selecting solutions and branches from the hierarchy and display business processes, interfaces, and executables, for example.
* Starting reports for a selected branch, to evaluate technical bills of materials (TBOMS) and test cases, for example.
* Creating test cases

For more information, see [Solution Documentation](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/3ef70f5452e7024be10000000a44176d.html).

## Activities

**Creating Test Cases**

In SAP Solution Manager 7.2, you create test cases for a business process step in a solution.

1. In the **Test Suite - Test Preparation** application, open a solution documentation.
2. Drill down to the business process step, for example, for which you want to create a test case.

In the table below, you see the elements that are included, for example executables and test cases that have been assigned already.

Right-click into the table with elements and choose **New**  **Test Cases**. Then you can choose from the following options:

* + **Assigning an existing test configuration.**

When you choose this option, you have to select a test configuration.

Then you define further information for the test case:

* + - Responsible person
    - Test classification, such as the following:
      * Functional integration test
      * Generic functional test
      * Regression test
      * Single functional test
      * User acceptance test
    - Test data variant
    - Search terms: This can be helpful if you have many test cases and want to assign further characteristics, for which you can search in the quick criteria maintenance. Search terms are not defined in the solution but in the test configuration.
    - Target logical components
  + **Creating a new test configuration.**

With this option, you can create a new test configuration and record a new test script (or assign an existing test script. When you choose this option, the editor for test configurations opens, and the popup in which you define the header data. For more information, see [Creating Test Configurations and Test Scripts](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/fb9bf93de5534a43862558adf678e5bc.html).

When you have completed the test configuration it is assigned to the selected business process step.

Then you define the characteristics for the test case as described above.

* + **Creating a new test document for manual testing, from a template.**

Also for test documents, you also define further test case characteristics:

* + - Status
    - Sensitivity
    - Priority
    - Keywords
    - Owner
    - Responsible
    - Test classification (see description above)
    - Duration
  + **Uploading an existing test document for manual testing.**

You can upload any local text file with a written description which guides the tester through a test case.

* + **Assigning an existing test document for manual testing.**

You can assign the same document that has already been uploaded for another test case, or you can copy an existing document to create an independent version.

1. Newly created test cases appear on the element tab for the business process.
2. Save and close the window.

You can now assign the test cases to a test package, under **Test Plan Management**. For more information, see [Test Plan Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/48e151fddb163184e10000000a421937.html).

**Test Case Types**

You can use the following test case types when you create a test:

* **Test Document**: refers to manual tests.
* **Test Configuration**: refers to automatic tests for all automatic tools that are registered in your system. (Tools for composite tests and the eCATT test tool do not need to be registered explicitly.)

Note

You may want to deactivate certain test case types, for example, because the license of a third-party test tool has expired.

In this case you would cancel the registration of the respective third-party test tool. For more information, see [Configuration of Test Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/572ede554cf5d207e10000000a44538d.html).

# Test Execution Reports

There are the following applications for test reporting in the SAP Solution Manager launchpad:

## Test Suite - Overview

In the **Test Suite - Overview** application, the following tabs present the status of manual and automatic testing:

* **Test Planning**

Shows the test plans and the number of test cases, executables, manual and automatic test, open test cases and open defects. When you select a test plan, the status of the included test packages and test cases is shown.

* **Test Execution**

Shows the status of the users own test packages in a test plan.

## Test Execution - Jobs

This list contains the background jobs that were created by the scheduling of automatic tests in the in tester worklist.

Two queries are available: **My Jobs** and **All Jobs**. You can perform complex searches by defining new queries, or change the selection criteria of existing queries.

You can navigate to the job details by clicking on the job name.

## Automated Tests - Results

This application shows the logs of executed automatic tests.

In the log overview you can define your search criteria as required to check on the status of your executed automatic tests.

You can navigate to single log details by clicking on the log ID. This view shows more detailed information on:

* Execution Status
* Execution Context
* Solution Manager Context

so you can investigate test execution errors, for example.

# Test Suite - Analytics

## Use

In the **Test Suite - Analytics** application of the test suite, you can use reports to evaluate information on testing.

## Prerequisites

You have defined the aggregation rules to calculate the test case status, for test plans and test packages, in the customizing for SAP Solution Manager, under **Capabilities**  **Test Management**  **Test Suite for SAP Solution Manager**  **Preparation**  **Setup**  **Central Test Suite Settings**  **Customize Test Plan Evaluation**.

## Activities

1. On the Launchpad, choose the **Test Suite - Analytics** application.
2. Select the report you want to execute from the following groups:
   * [Completeness and Gap Reports](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/f8b31b5684b77c65e10000000a44147b.html)
   * [Test Execution Analytics](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/4cb41b5684b77c65e10000000a44147b.html)
   * [Status and Progress Analytics (BW-Based)](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/83a7cd5639d26c28e10000000a4450e5.html)
3. A new window opens, where you can define your search criteria and get the corresponding results.

# Test Execution Analytics

# Use

You can perform status analyses for one or several test plans or test packages, with different types of analyses.

## Features

You can start the reports from the launchpad, in the **Test Suite** launchpad group. The following reports are available under **Test Execution Analytics**:

**Multiple Test Plan Status**

In your search results, view all test plans corresponding to your search criteria, with aggregated information on test execution. For example, view the number of test cases in progress (with statuses of no result, errors, OK), and the number of open defects.

For the overall test plan, view the displayed test plan status, person responsible, number of test packages and testers, and the number of unassigned test cases.

When you select a test plan, several graphics illustrate the status:

* Test cases per type
* Test cases per status
* Defects per priority

You can also access the test case analysis for the selected test plan.

**Multiple Test Plan Status Details**

The test plans listet in your search results give you detailed information about the test execution status of test packages, test cases and testers.

**Test Plan - Test Package Analysis**

The search result shows a hierarchy with the following levels: Test plan, test packages, and assigned testers.

For the test plan and test package rows, you can see aggregated values about test evaluation, for example the number of involved testers, number of test case per specific status, number of defects, and the number of test cases per type.

For tester row, you can see the business partner of the tester and whether the partner's assignment is locked or unlocked. Choose **Go to** to navigate to the other reports available for the selected line.

Choose **Personalization** to add additional columns. In the hierarchy, the node description is a link with which you can navigate to the relevant object, for example to the test plan or test package.

**Test Plan - Test Cases Analysis**

The search result shows the test plan hierarchy, with status information. On test plan level, view aggregated values about test cases execution status: Number of unassigned test cases, number of test cases per status, and number of defects.

On test case level, you can see individual status information, such as date and last tester, and some test case data, such as the test case classification and priority.

Choose **Personalization** to add additional columns. In the hierarchy, the node description is a link with which you can navigate to the relevant object.

Click on the status icon of a test case, to open a popup with all test statuses per package and tester. These statuses are aggregated to the overall test case status.

Click on the test note or test result icon to access the **Test Documents Overview** in a new tab. This is where you can view and sign your test note or test result.

**Test Package - Test Cases Analysis**

The search result shows the test package hierarchy, with status information. On test package level, you see aggregated values about test cases execution status: Percentage of results still missing, test cases per status, and number of defects.

On test case level, you can see individual status information, the date and last tester, and for example the test case classification and priority.

Choose **Personalization** to add additional columns. In the hierarchy, the node description is a link with which you can navigate to the relevant object.

Click on the status icon for a test case, to open a dialog box with all test statuses per package and tester. These statuses are aggregated to the overall test case status.

**Test Plan - Defects Analysis**

This report gives an overview of the defects that were created in a test plan. Testers create defects if a test could not be performed.

1. You can define several criteria to get an analysis for all test plans.
   * To determine the granularity of the report, use the following lists:
     + **Defect Analysis**: Details about the defect and the related test case. So if a defect has been assigned to more than one test case, it appears several times in the list.
     + **Defect Overview**: Defect information in an aggregated mode, so that each defect only appears once.
   * You can select defect transaction types.
2. Depending on the search criteria, the system displays an analysis or an overview of the defects for the test plans, including useful information such as the defect ID, status, priority, and reporter.
3. To open the incident for a defect in your incident management system, click the defect ID. Here you can get more information and process the incident in [IT Service Management](https://help.sap.com/viewer/fbc7b5ecf5094fe0b6a2eb966160008f/7.2.07/en-US/d6b729db0a2a41589e9fa8f25eba7fcb.html).

Defect information is directly retrieved from the incident system, and therefore continuously up to date.

Alternatively, you can access reports a test plans in the **Test Plan Management** application. Select a test plan and choose **Go To**  **Defects Overview**.

**Test Package - Defects Analysis**

This report shows the defects that were created for test cases in one test package. It is similar to the defects analysis for test cases, but only shows the defects for a selected test package.

**Test Result Document**

With this report, you can generate a Microsoft Word document with information on a test plan. This comprises test package information, defects, and landscape details.

Select a test plan and the information you want to have in the document. Then choose **Execute** and save the file locally.

**Traceability Matrix**

With the Traceability Matrix report, you can trace the whole lifecycle of your projects, IT requirements, change requests and change documents, as well as transport loops, which the projects undergo until their fulfillment. Typical users for this report are project managers, quality managers, and test managers.

After you have set a filter on your projects, the report will trace starting from the projects to the relevant processes and display the relevant results.

You can find the KPI definition and calculation in the KPI Catalog.

The traced processes are as follows:

* PPM project
* IT requirement
* Business requirement
* Request for change
* Change document
* Test plan, test package, test case
* Defect
* Transport
* Change cycle
* Solution document

**Prerequisite**

In **SAP Solution Manager Configuration**, complete the following steps:

* Configuration of Project Management
* Configuration of IT Service Management
* Configuration of Change Control Management
* Configuration of Test Suite
* Configuration of Dashboard Builder