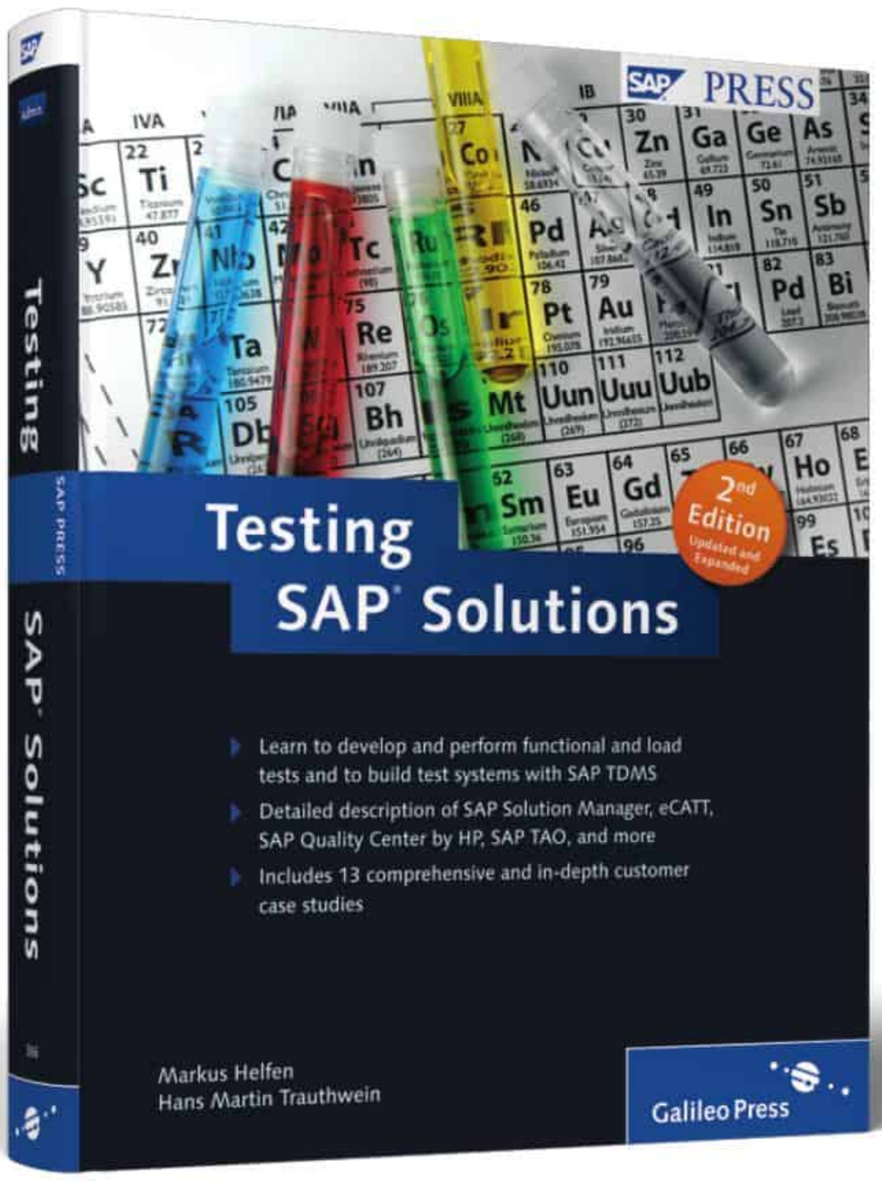
Markus Helfen and Hans Martin Trauthwein

Testing SAP® Solutions



Galileo Press

Bonn • Boston

366 Book_M.indb 3 12/7/10 3:20:46 PM

Contents at a Glance

Contents

366 Book_M.indb 7 12/7/10 3:20:46 PM

366 Book_M.indb 8 12/7/10 3:20:46 PM

366 Book_M.indb 9 12/7/10 3:20:46 PM

366 Book_M.indb 10 12/7/10 3:20:46 PM

366 Book_M.indb 11 12/7/10 3:20:47 PM

366 Book_M.indb 12 12/7/10 3:20:47 PM

366 Book_M.indb 13 12/7/10 3:20:47 PM

366 Book_M.indb 14 12/7/10 3:20:47 PM

8.2 Creating and Running UI-Driven Tests

Using eCATT, you can test the following UI-based applications: transactions with and without controls as well as Web Dynpro applications.

8.2.1 Testing Transactions without Controls (TCD)

To create an eCATT script, first start eCATT (Transaction SECATT). In the initial screen (see Figure 8.11), select the TEST SCRIPT item, and enter a name. Note that the name must reside in the customer-specific namespace.

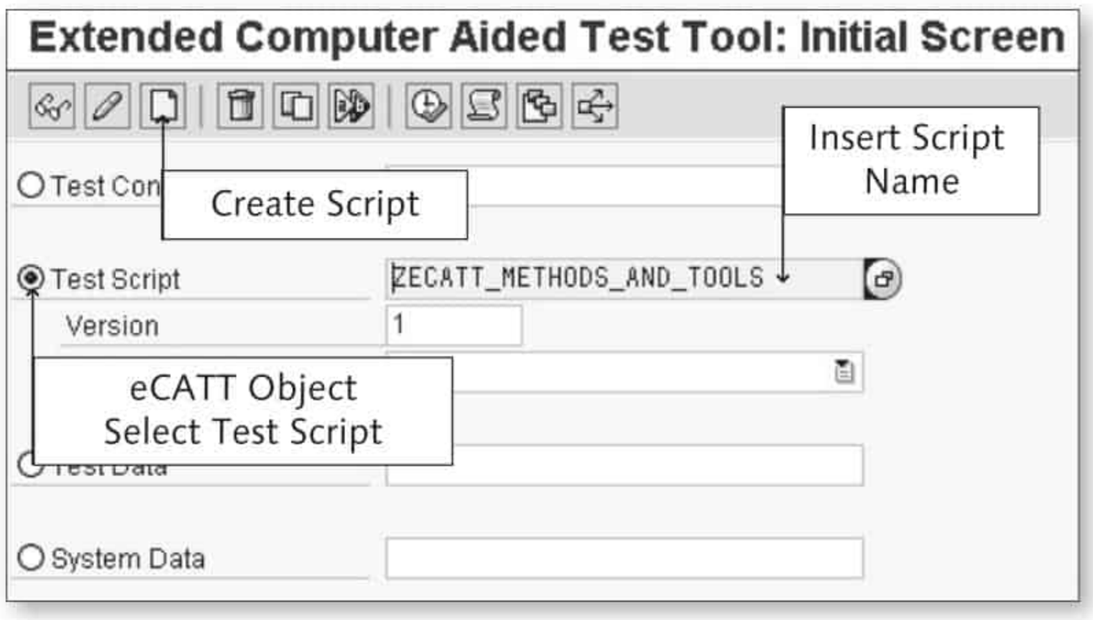


Figure 8.11 eCATT Initial Screen

Script editor

The script editor is the central tool for creating eCATT scripts. Its interface is divided into three areas (see Figure 8.12). The lower area is occupied by the command editor. There you edit the script logic that is composed of the different commands.

The upper area provides an input possibility for creating the parameters of a script. If you want to edit a command interface, the structure editor is opened to the right. It can be enabled by double-clicking on the name of a command interface in the command editor.

366 Book_M.indb 426 12/7/10 3:22:52 PM

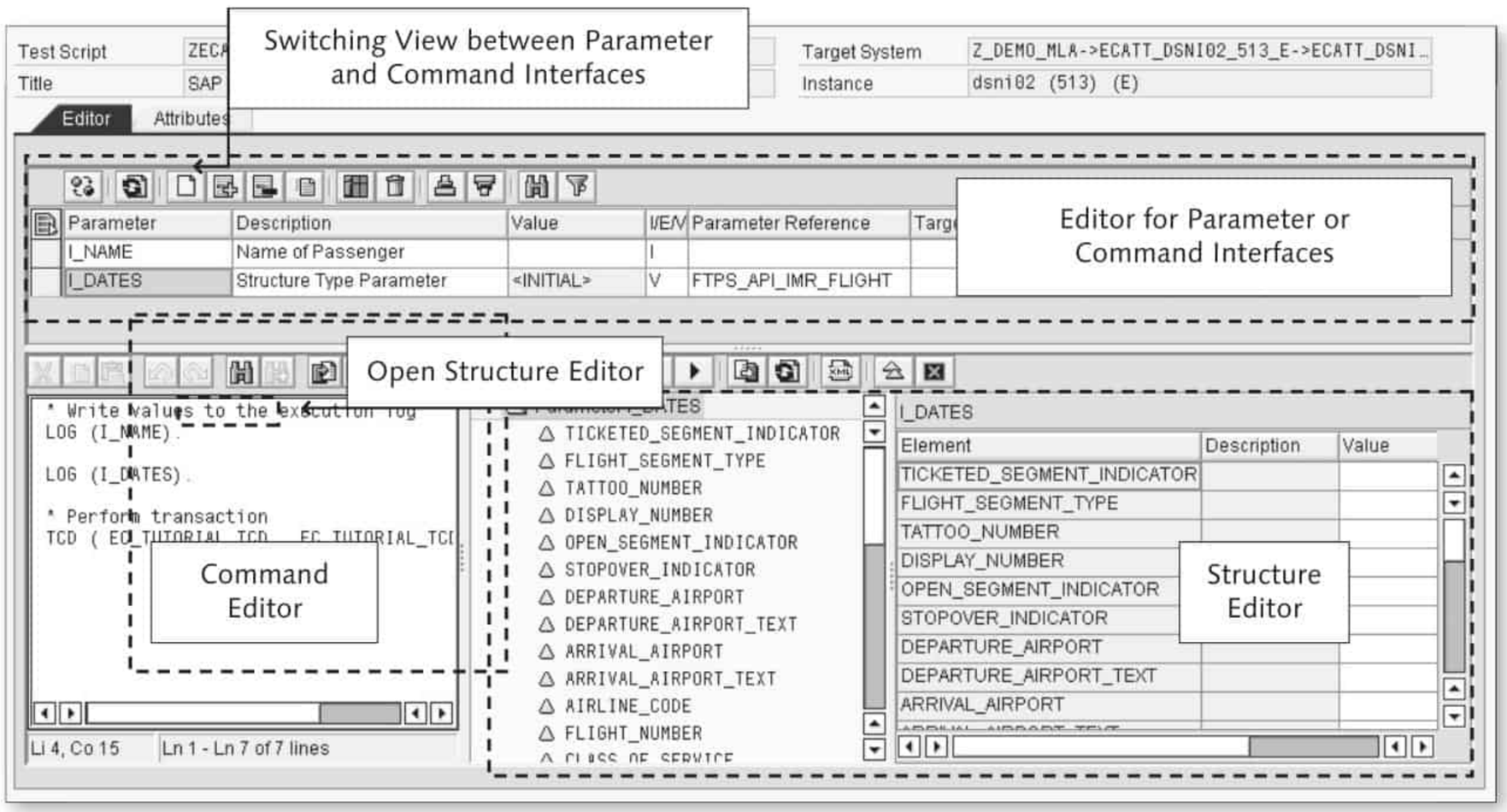


Figure 8.12 Script Editor

The first step in creating the script logic is recording a transaction. Use the RECORDER function of the script editor for this purpose.

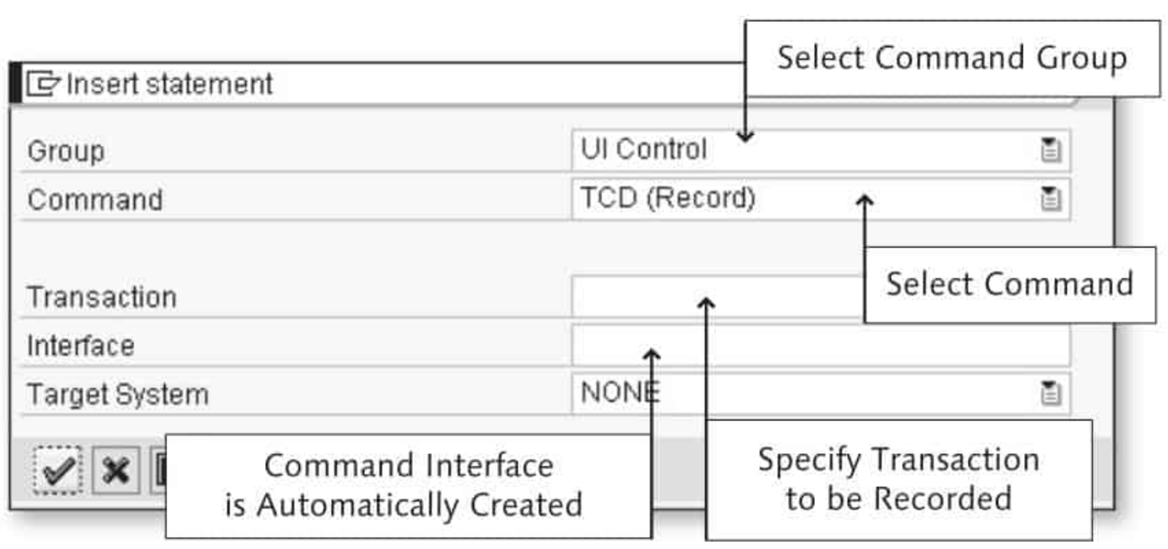


Figure 8.13 "Start recording" Dialog

Start recording

To start recording, in the script editor select the PATTERN button. A dialog is displayed where you can specify the necessary settings for the recording (see Figure 8.13). The available commands are sorted by functions and divided into groups. First select the desired UI ADDRESSING function group and then the test driver. In the following description, we assume that you use the TCD driver for recording transactions without

366 Book_M.indb 427 12/7/10 3:22:57 PM

controls. As soon as you select the driver, you can select the transaction to be tested. An appropriate interface is then automatically created by the script editor.

eCATT now opens the transaction. Perform the transaction as usual and then close the transaction window F12. eCATT asks you if you want to accept the data. If you confirm with OK, a new command line with the TCD command is displayed in the editor. The entire recorded transaction is now included in the corresponding command interface.

"TCD" Command

The TCD command is designed for addressing the TCD driver. It has the following format:

```
TCD ( <transaction code>, <command interface>,
      [<target system>] ).
```

A TCD command must always be created via a recording.

Command interface

When a TCD driver is used, the command interface records all dynpros with all fields that have been displayed in the transaction (see Figure 8.14). The data you entered and the default values for fields you did not fill in are recorded.

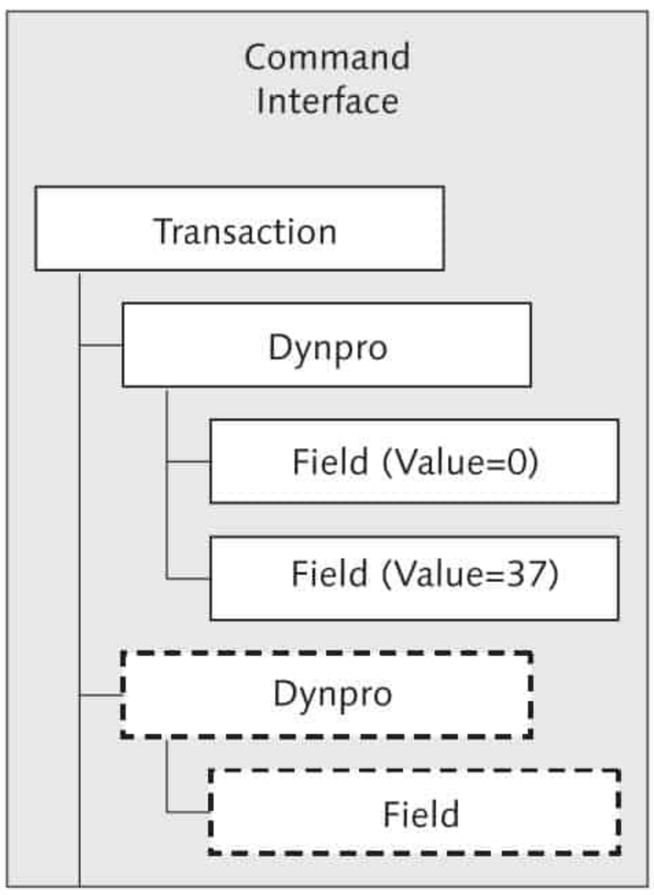


Figure 8.14 Structure of the Command Interface

366 Book_M.indb 428 12/7/10 3:22:57 PM

The values you entered are displayed as fixed values in the command interface. You can use the search function later to find these fixed values during parameterization. No fixed values are recorded for fields you kept at their preset default values. During parameterization, you therefore you might have difficulties with identifying these fields. When recording, ensure that you enter input values for all fields that are relevant to the test case.

Script parameterization

If the test case is to be executed with values other than those used during the recording, the corresponding fixed values need to be replaced with parameters (see Figure 8.15). During execution of the test script, the parameter value is then inserted in the appropriate place. This procedure links a part of the command interface of the `TCD` command to parameters that are visible from the outside. In other words, fields of the command interface can be linked to parameters to return results of the `TCD` command. Use import parameters to transfer values to input fields. Use export parameters or local variables to process results from the command interface.

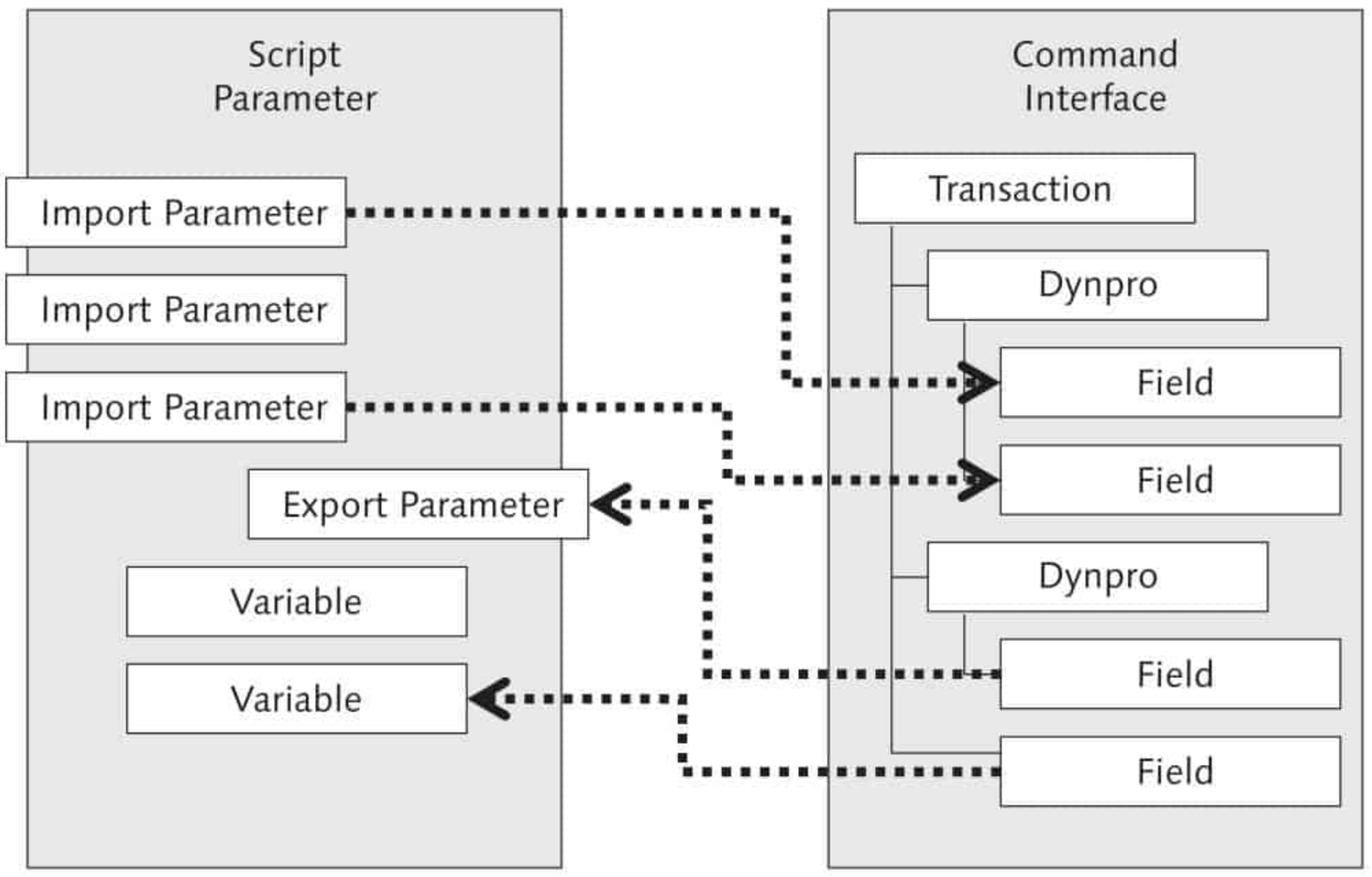


Figure 8.15 Replacing Fixed Values Entered with Parameters

The automatic creation of parameters is supported as of SAP Web AS Release 6.40. You can also manually create the parameters required. For this purpose, complete a row in the parameter editor to add a parameter (see Figure 8.16).

366 Book_M.indb 429 12/7/10 3:22:58 PM

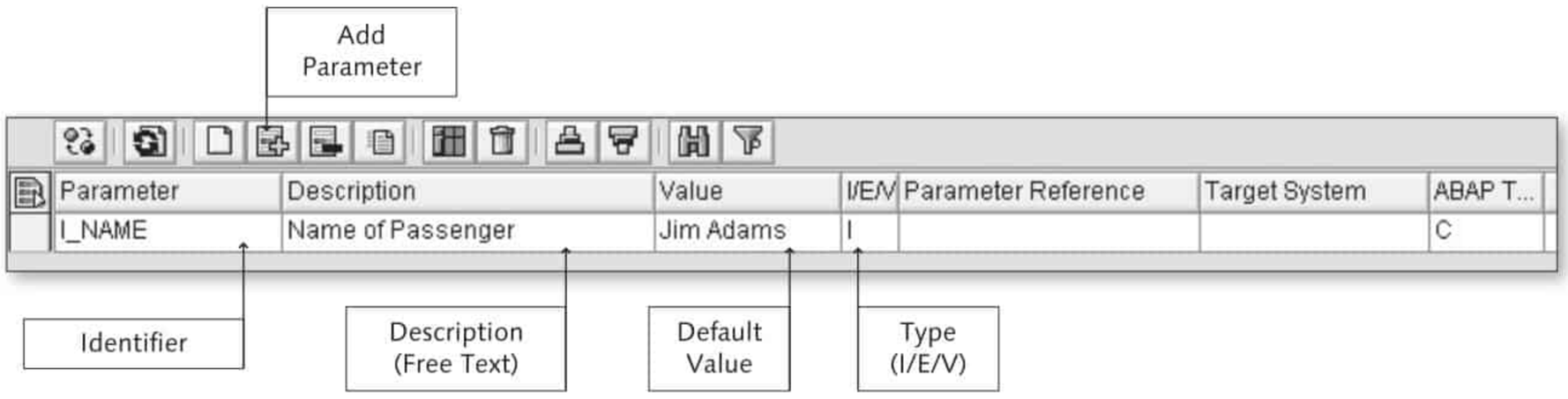


Figure 8.16 Adding a Parameter

Recommended procedure for naming conventions

Every parameter has a unique name for identification. To improve the readability of the scripts, we recommend keeping a consistent notation. A well-established procedure is to have all import parameters start with "I_," all export parameters with "E_," and all local variables with "V_."

Parameter types

Once the names of the parameters have been defined, you specify their visibility. In the parameter editor, specify "I" for import or "E" for export parameters. If you need local variables you can create them now as well. Set the type to "V."

To parameterize a recorded `TCD` command, open the related command interface in the structure editor (double-click). Look for the entered fixed values. When you have found the corresponding field, replace the value with the name of the parameter (see Figure 8.17).

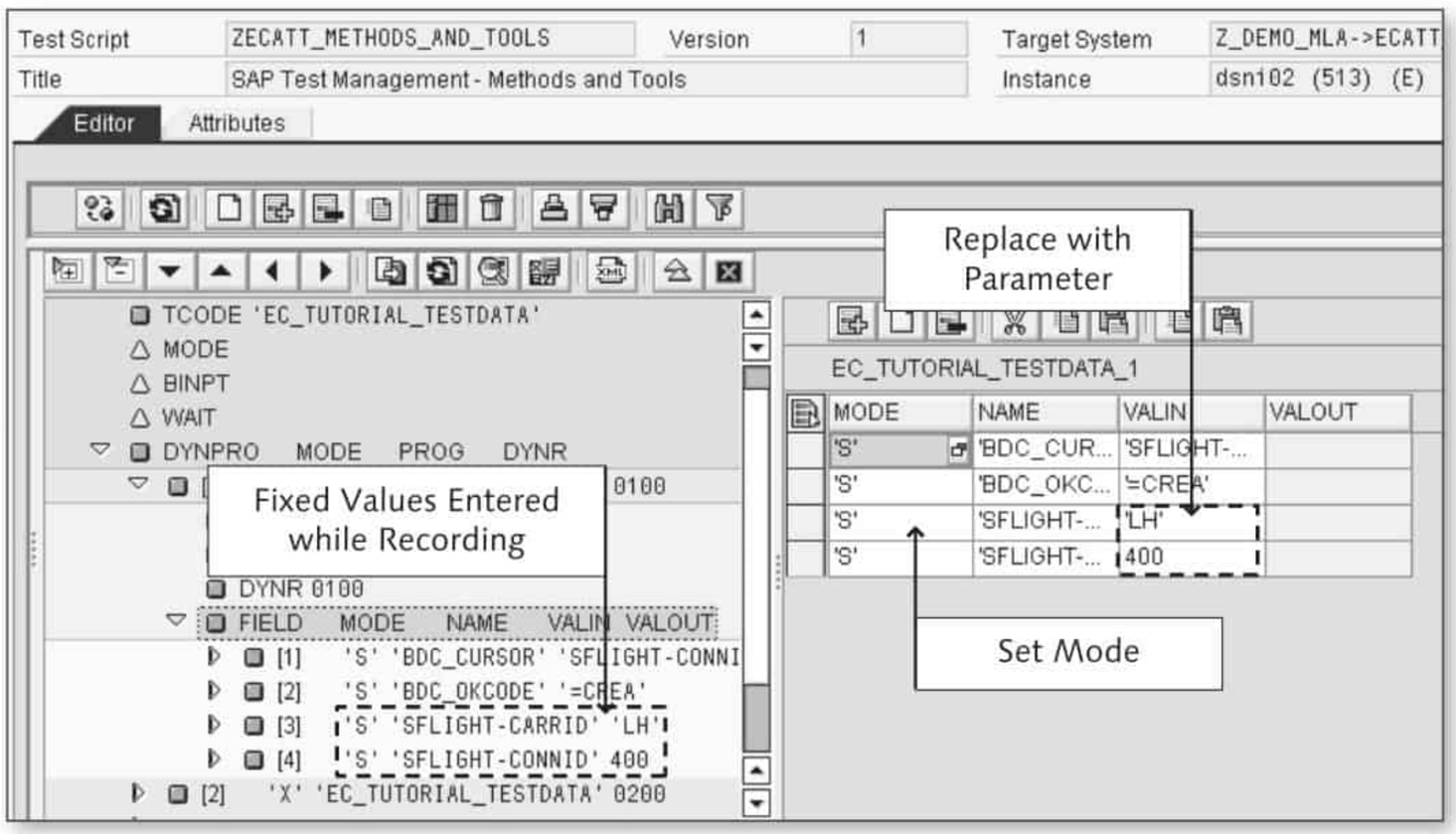


Figure 8.17 Script Editor During Parameterization

366 Book_M.indb 430 12/7/10 3:23:01 PM

Parameter mode

eCATT supports different kinds of parameterization. We distinguish between three actions. Two passive modes are also available. In the MODE column, you can set the appropriate mode:

1. **S (set value)**
   This mode is used for transferring import parameters or local variables to transaction fields. If you select this mode, the parameter value is transferred to the corresponding dynpro field during script execution, just like user input.
2. **G (get value)**
   This mode is designed for exporting values from the command interface. A value calculated by the transaction is transferred to an export parameter or a local variable after the `TCD` command has been executed.
3. **C (check value)**
   The third mode permits a simple verification of the results. The value returned by the `TCD` command is compared to the specified parameter. If the comparison fails, the test case is regarded as faulty. Note that the possibilities of this test are rather limited. For more complex conditions, you should use the "G" mode to first read the field value and then check it later using the `CHEVAR` command. More information on this topic can be found in Section 8.8, *Modularizing Test Scripts*.
4. **I/O (passive)**
   The "I" mode refers to an input field that is not changed by eCATT. Via user parameters, the application can predefine the field with values, though.

   The "O" refers to an output field that is neither read nor checked by the eCATT script.

Dynpro simulator

The dynpro simulator is an alternative to the value maintenance in the field lists. To open the simulator, select the desired dynpro in the list of recorded dynpros and then the SIMULATE DYNPRO function.

The dynpro of the application is simulated on the screen and provided with eCATT-specific functions; for instance, INSERT PARAMETER. You can also navigate between the recorded dynpros so that you can edit the entire recording.

366 Book_M.indb 431 12/7/10 3:23:02 PM

The relevant values, parameters, and mode entries are copied to the field list when you exit the dynpro simulator.

Re-recording

The TCD driver enables you to re-record driver calls that have already been parameterized. This option is used if the test script encounters an error after the underlying transaction has been changed; for example, after installing a support package. Such an error can have two causes. The obvious cause is that the change has caused an error in the application logic. In this case, the error must be corrected in the application. Although the joy at a successful test—testing is always successful when errors are found—is usually muted by the awareness of difficulties for the operation or the project, valuable insights can still be gained.

The second possible error source consists of an incompatible change in the structure of the recorded transaction for the existing test script. Because the structure of a transaction changes more frequently than its fields (a field can be assigned to a different dynpro or another group but is hardly ever renamed or deleted), you can re-record an existing TCD driver call while maintaining the parameterization.

Double-click on the command to open the CHANGE COMMAND dialog (see Figure 8.18). Trigger the re-recording of the transaction, and record the transaction as usual. Fields that have already been parameterized are taken over from the old command interface according to their field names. In most cases, the transaction is already fully parameterized immediately after the recording. Only newly added fields must be completed during the parameterization. This functionality is exclusively available in the TCD driver and makes the recordings performed using this driver extremely easy to maintain.

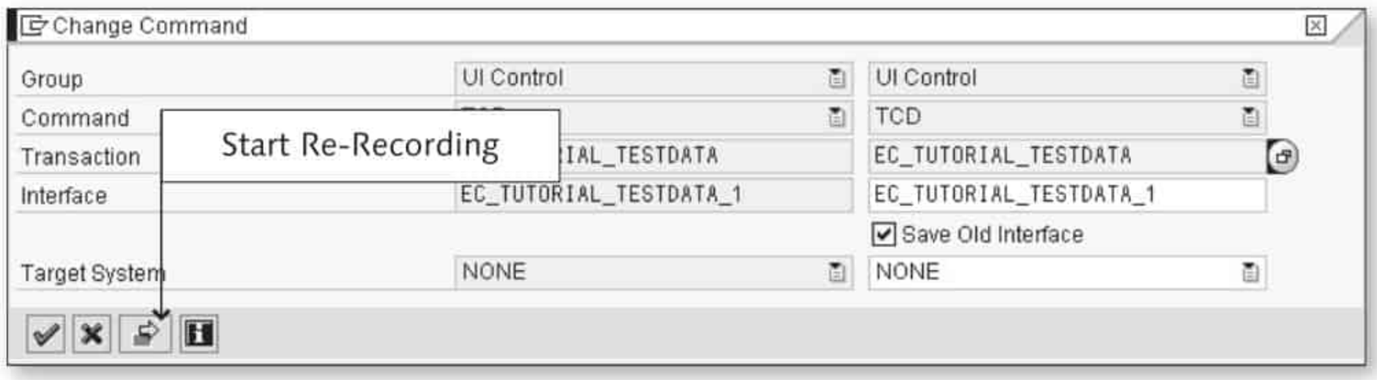


Figure 8.18 "Change Command" Dialog Window

366 Book_M.indb 432 12/7/10 3:23:03 PM

Start options for the TCD driver

For processing a script, you are provided with various start options for selection. The options relevant to the TCD driver can be found under the UI CONTROL tab in the TCD area (see Figure 8.19).

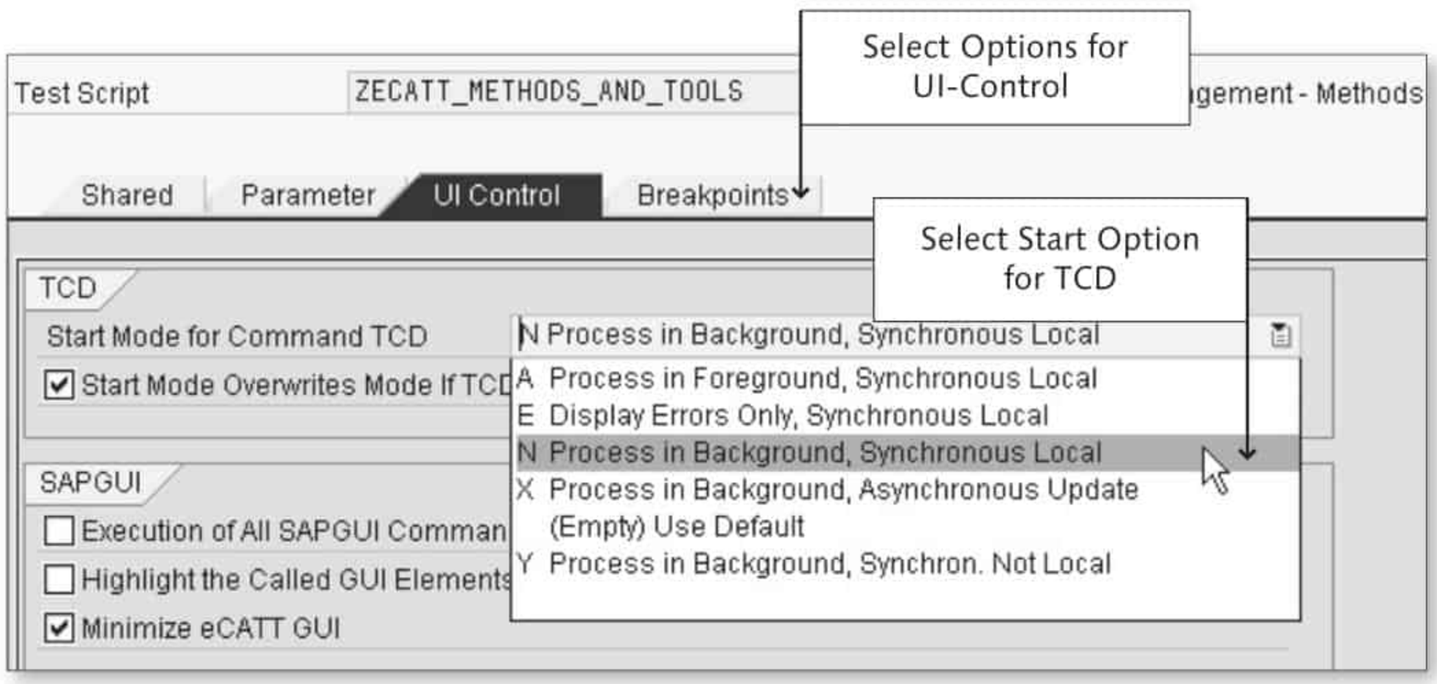


Figure 8.19 Start Options for Processing a TCD Recording

Start options are selected before a script is processed. You have the following options:

- **Process in Foreground, Synchronous Local**
  The script is processed in the foreground; that is, with a user interface. All actions of the script can be observed on screen. The database is updated synchronously. This option ensures that all input values have been updated in the database before the next step in the script is executed. For tests using eCATT, you should always select one of the options with synchronous update for the reason mentioned above.
- **Display Errors Only, Synchronous Local**
  This option processes the script in the background; that is, without a user interface. If an error occurs during the execution, the corresponding position is displayed in the user interface. The error can now be manually corrected. The script is then continued in the background until another error occurs or the test case has been completed. The database is updated synchronously.
- **Process in Background, Synchronous Local**
  This option processes the script completely in the background. There is no output to the screen. Errors are not reported immediately, but they can be viewed in the automatically created log of the eCATT run

366 Book_M.indb 433 12/7/10 3:23:05 PM

after the script has been executed. The database is updated synchronously.

- **Process in Background, Asynchronous Update**
  This option processes the script completely in the background. The database is updated asynchronously. This means that the control flow might return to the script before the updater has changed all values in the database. For a subsequent step of the script, therefore, there is no guarantee that a change from a previous step has been implemented in the database.

  The option can be used when eCATT is implemented to provide mass data in the system. This can happen either during a data migration or when test data is created for preparing manual tests or trainings. By disabling a synchronous update, you can often considerably increase the speed.

- **Process in Background, Synchronous Not Local**
  The script is processed in the background, and the update takes place synchronously but via a different work process than the transaction. This option is obsolete and is only supported for downward compatibility.

- **Use Default**
  The option stored in the command interface of the command is used.

Regarding the message handling within the TCD driver, please refer to Section 8.6.3, *Message Handling*.

8.2.2 Testing Transactions with Controls (SAP GUI)

Starting with SAP basis Release 4.5B, transactions are able to present more complex and user-friendly graphical user interfaces via SAP GUI controls. One characteristic of this way of programming is the requirement that a part of the application logic is run on the front end.

Because the TCD driver immediately interferes with the application server, application parts running on the front end are outside its reach. Therefore, eCATT provides its own test driver for recording transactions with controls. The SAPGUI driver works with the SAP GUI for Windows and interferes with the SAP system at a different level than does the

366 Book_M.indb 434 12/7/10 3:23:05 PM

TCD driver. An important difference is that the SAPGUI driver does not connect to the application server but to the front end.

"SAPGUI" Command

The `SAPGUI` command is designed for addressing the SAPGUI driver. It has the following format:

```
SAPGUI ( <command interface>, [<target system>] ).
```

In contrast to the `TCD` command, the `SAPGUI` command only allows the transfer of values to the application. There are separate commands for reading and testing values.

Events

While the TCD driver records the result of the input in the record fields, the SAPGUI driver registers events. These events refer to changes of the state of screen elements; for instance, the selection of a value in a listbox, the input of text in a text field, or the expansion of a branch in a tree control. In particular, this means that the SAPGUI driver only records information about the fields that the user actually changes during recording. Another difference is that the SAPGUI driver uses different commands for reading and querying screen elements while the TCD driver serves all actions of the `TCD` command (see Table 8.2).

Function	TCD Driver	SAPGUI Driver
Parameterizing	TCD (mode S)	SAPGUI
Reading	TCD (mode G)	GETGUI (as of SAP Web AS 6.40)
Check	TCD (mode C)	CHEGUI (as of SAP Web AS 6.40)
Passive (output)	TCD (mode O)	
Passive (input)	TCD (mode I)	

Table 8.2 Commands for Parameterizing, Reading, and Checking Field Values

Recording granularity

Because this type of recording generates many events for complex transactions you must specify the appropriate level of granularity prior to recording. This level of granularity determines the number of individual commands into which the script is subdivided. The higher the level of granularity for the recording, the better the overview, reusability, and maintainability of the script. It is also easier to insert descriptive comments between the individual steps. Figure 8.20 shows the different

366 Book_M.indb 435 12/7/10 3:23:05 PM

levels of granularity, with the level of granularity increasing the farther you go to the right.

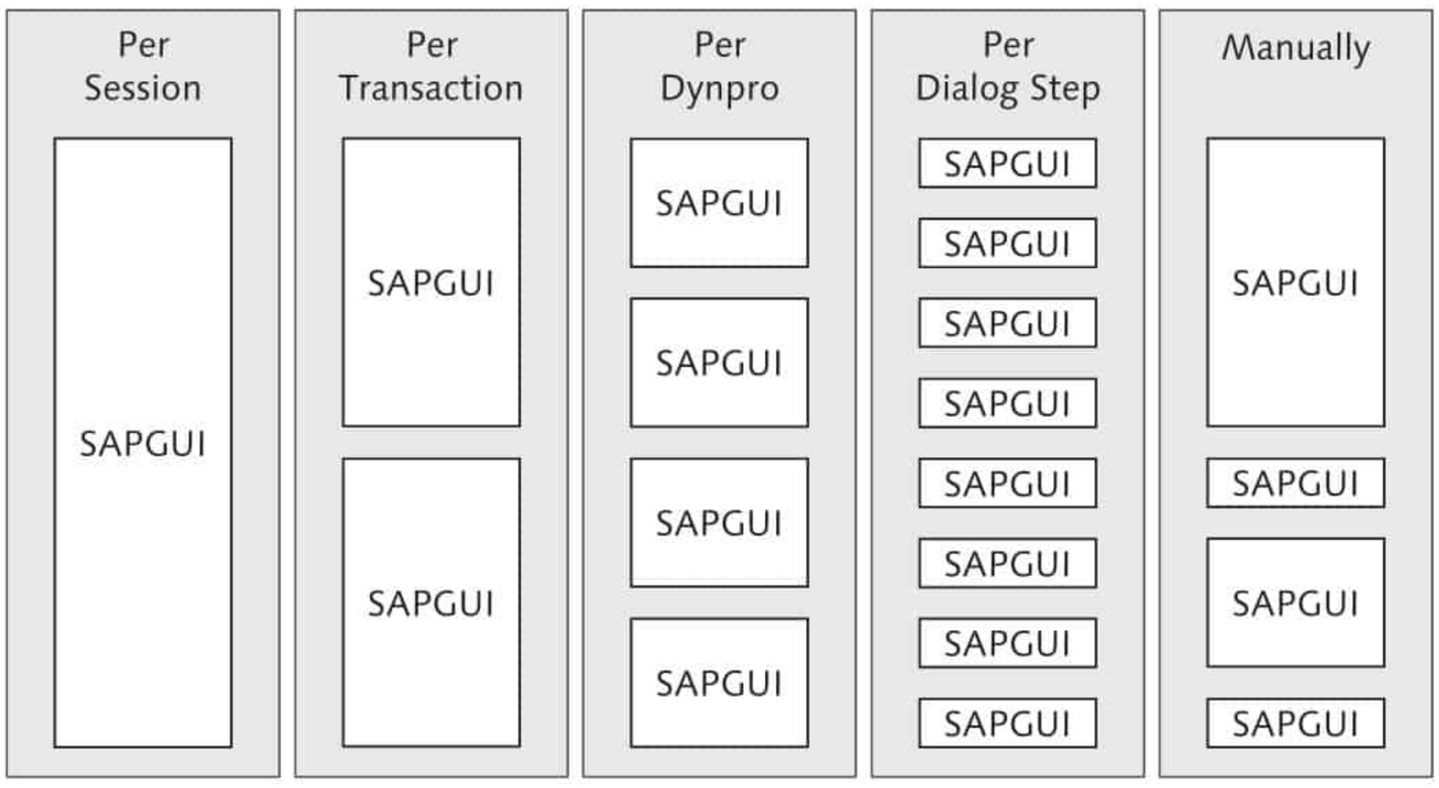


Figure 8.20 Granularity Levels of the SAP GUI Recording

The various granularity levels of the SAP GUI recording have the following meanings:

1. **Per dialog step**
   For every GUI event (every roundtrip between front end and back end), a separate row containing one `SAPGUI` command is inserted into the script.
2. **Per dynpro**
   Events referring to the same dynpro are joined to form one command.
3. **Per transaction**
   Events referring to the same transaction are joined, even if they span several dynpros.
4. **Per session**
   All events between starting and ending an SAP GUI session are joined to form one command.
5. **Manual**
   The creation of an `SAPGUI` command is explicitly triggered by the user, who activates an appropriate button during the recording. If you use

366 Book_M.indb 436 12/7/10 3:23:06 PM

this option, you should add a comment to the creation of every command to maintain a better overview.

In general, the granularity level per dynpro as proven as the most appropriate solution for most requirements.

If you want to change the granularity of your script subsequently, you can join steps using the `Join` command or split them using `Split`. If it should become necessary to change the test script, for example, to adapt it to a new support package level, you can use the `SAPGUI (Attach)` command to re-record the affected part of the script, rather than the entire sequence. These commands are described in more detail in the course of this chapter.

Prepare recording

To prepare recording of one or several `SAPGUI` commands, go to Transaction SECATT, create a new test script, and then select the PATTERN button. In the INSERT STATEMENT dialog window, select the UI CONTROL group and then the SAPGUI (RECORD) command. If you leave the [GENERATED] presetting in the INTERFACE field unchanged, eCATT generates a name for the command interface based on the selected granularity. For example, if you selected the granularity level per transaction, the generated name includes the transaction code followed by a number. If the generated name does not match your naming concept, you can enter the desired name in the INTERFACE field.

When you confirm the entries, the system shows a dialog window in which you set the granularity of the recording (see Figure 8.21).

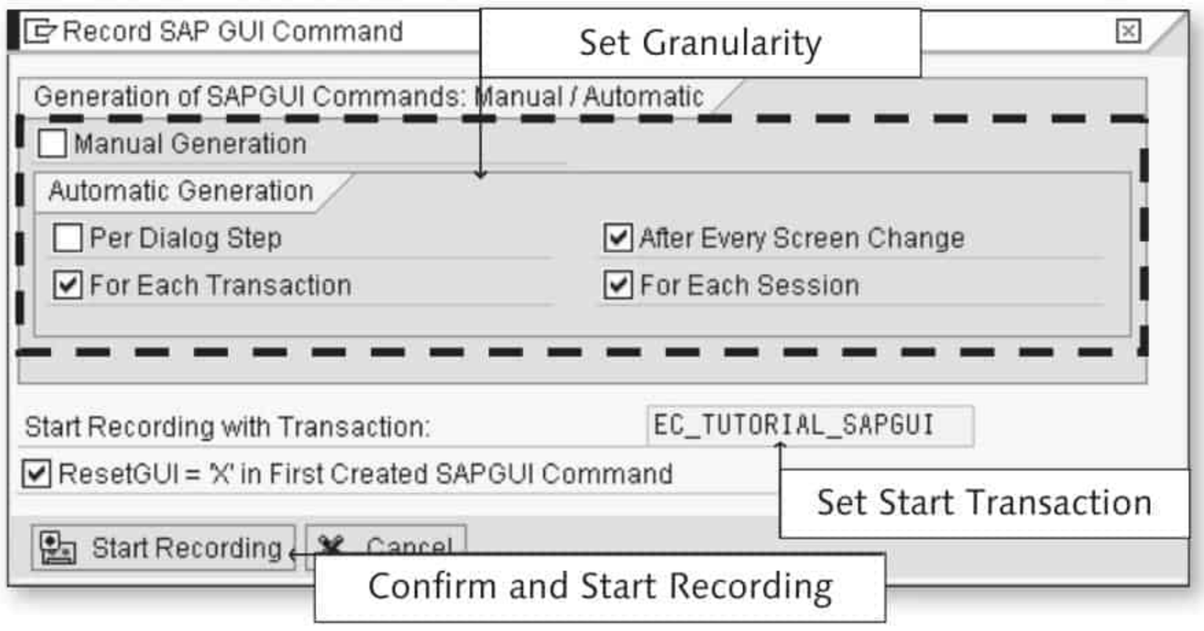


Figure 8.21 Setting the Granularity Levels of the SAP GUI Recording

366 Book_M.indb 437 12/7/10 3:23:07 PM

With SAP Web AS Release 6.40, you must specify a start transaction. Then, the recording starts directly with the first screen of the transaction selected. In SAP Web AS Release 6.20, by contrast, you must start the transaction by entering the appropriate transaction code during recording.

ResetGUI If you've selected the RESETGUI option, a `ResetGUI = 'X'` flag is generated in the first generated command interface of the recording. This means that the command has a similar effect during processing as an "/n" prefixed to the transaction code. To explain further: A correct processing of the SAP GUI commands is only ensured if every subsequent command starts precisely at the point where the previous command has stopped. This condition is met with everything that you're recording in one single step. But if you record a test script in several steps, this might not be the case. For this reason, you can select the `ResetGUI` flag in the first `SAPGUI` command of every transaction when you process an SAP GUI flow with several individual transaction changes.

After you've made your selection, click START RECORDING to confirm. The system displays the RECORD SAP GUI COMMAND window (see Figure 8.22).

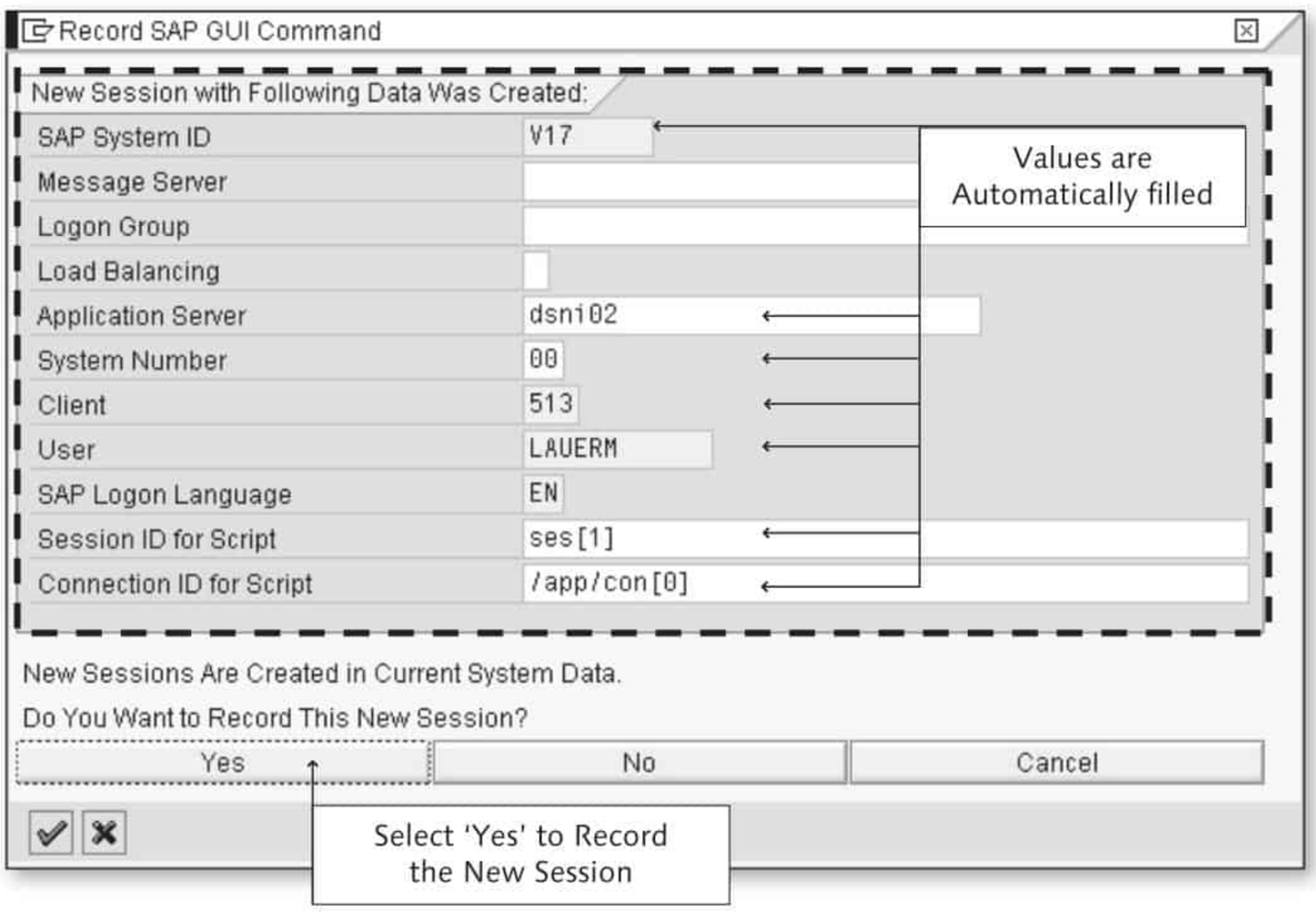


Figure 8.22 Dialog Window for Starting the Recording in a New Session

366 Book_M.indb 438 12/7/10 3:23:10 PM

eCATT generates a new session when a recording is started. It automatically recognizes the values of this newly generated session and displays them in the dialog window. In most cases, it is therefore sufficient to confirm the preset values with YES. SAP Note 1307732 provides further information on exceptions.

Session ID and connection ID

Based on the *connection ID* and the *session ID*, the GUI sessions are uniquely identified during the recording of `SAPGUI` commands. By default, these different IDs are also used for processing to address different sessions. *Session* always refers to a mode and *connection* to a target system. If all `SAPGUI` commands are supposed to be run in the same session, all `SAPGUI` command interfaces must have the same session ID and connection ID. Different combinations, in turn, stand for different sessions. This is significant if you work in different modes in parallel for a recording; for instance, if you record an application in one mode and open another mode in which information is to be checked and recorded at the same time. During processing, eCATT determines which command must be executed in which session, based on the different session IDs.

You have two ways to override these ID values. You can change values for multiple commands in the script editor simultaneously by selecting EDIT • PARAMETER/COMMANDO INTERFACES • REPLACE IDS IN SAPGUI INTERFACES. Alternatively, you can select the EXECUTION OF ALL SAPGUI COMMANDS IN A SINGLE SESSION PER DESTINATION option in the start options for all commands of a destination (that is, a target system). With this option, the different IDs are ignored and all SAP GUI commands are executed in the same session.

Run recording

If you confirm the values in the RECORD SAP GUI COMMAND window, the system displays the RECORDING RUNNING window (see Figure 8.23). In this window, you can implement different actions; for instance, inserting commands manually and stopping recording again. The functions of this window are described later on.

In parallel to the RECORDING RUNNING window, the start transaction for recording is opened in another mode, the recording mode (not in SAP Web AS 6.20, where you must still start the transaction manually). Execute the transaction as usual. If you selected the manual mode, you must change back to the recording window (see Figure 8.23) and trigger the creation of `SAPGUI` commands using the appropriate button. After the

366 Book_M.indb 439 12/7/10 3:23:10 PM

transaction has finished, go back the RECORDING RUNNING window and terminate the recording.

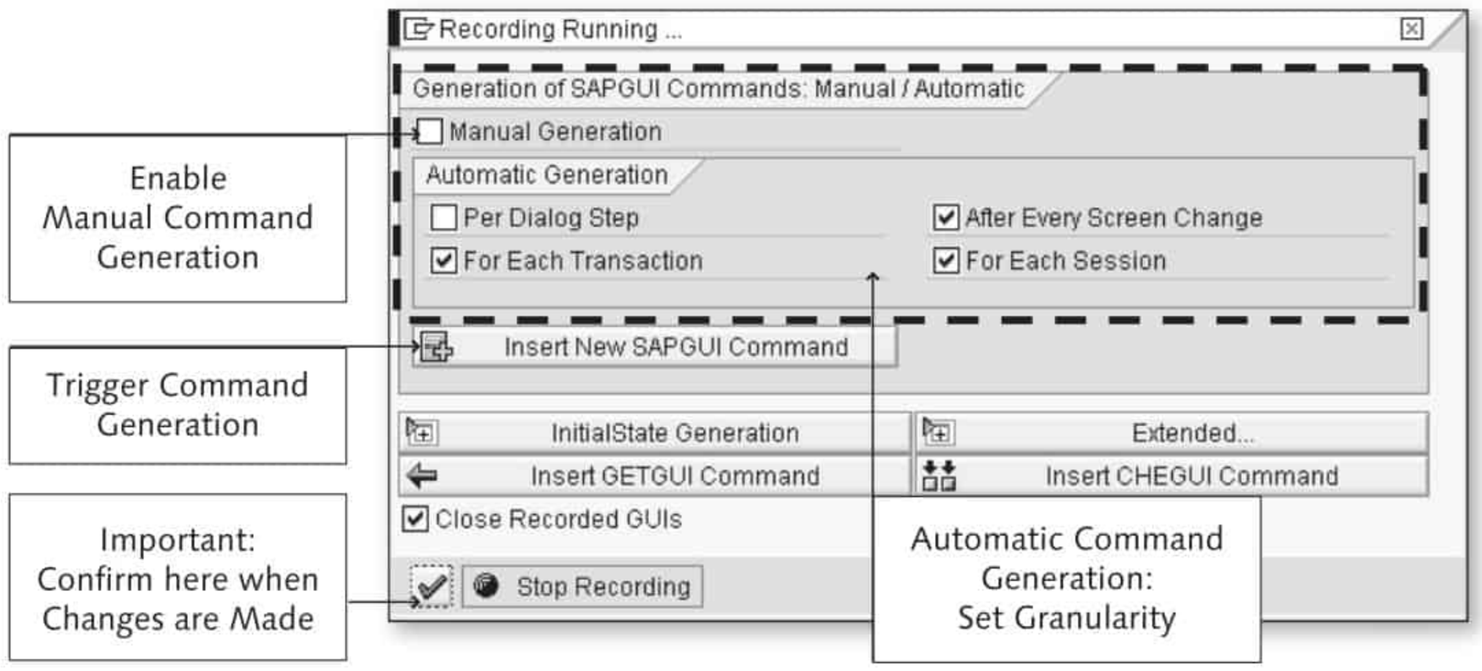


Figure 8.23 Dialog Window for Controlling the Recording

Using "CHEGUI" and "GETGUI"

In your scripts, you will often have to copy values from the screens or check contents. In eCATT, this is implemented with the commands, `GETGUI` and `CHEGUI`, which are available as of SAP Web AS 6.40. `GETGUI` reads the values of a GUI element, for instance, for a text field. `CHEGUI` reads and checks.

At the point where you want to determine a value, go back from the application mode to the recording dialog and select INSERT GETGUI COMMAND. If you want to additionally check the field content, select INSERT CHEGUI COMMAND (see Figure 8.24).

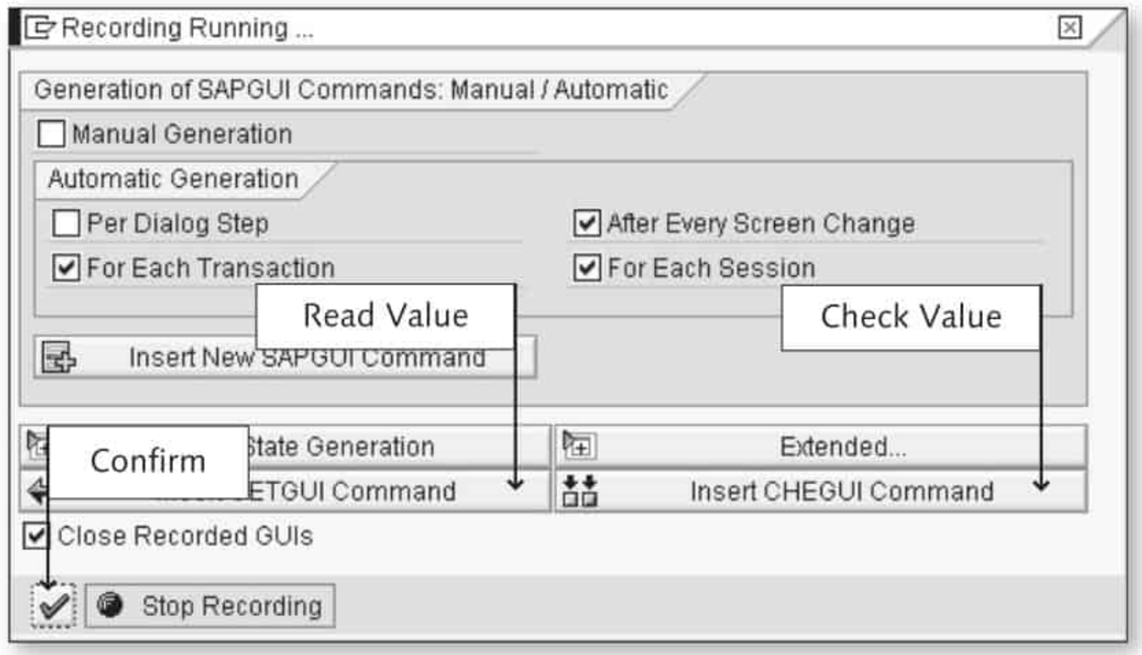


Figure 8.24 "Recording Running" Dialog Window

366 Book_M.indb 440 12/7/10 3:23:14 PM

After you've selected the relevant option, the control returns to application mode. The subsequent steps are identical for the two commands.

You can use the mouse to select the area whose value you want to determine. Because SAP GUI is in selection mode, the selected area is indicated with a colored frame if you move the mouse over it. The selected area is copied using the left mouse button.

After you have selected the corresponding field, the system takes you to the dialog for editing the `GETGUI/CHEGUI` command (see Figure 8.25).

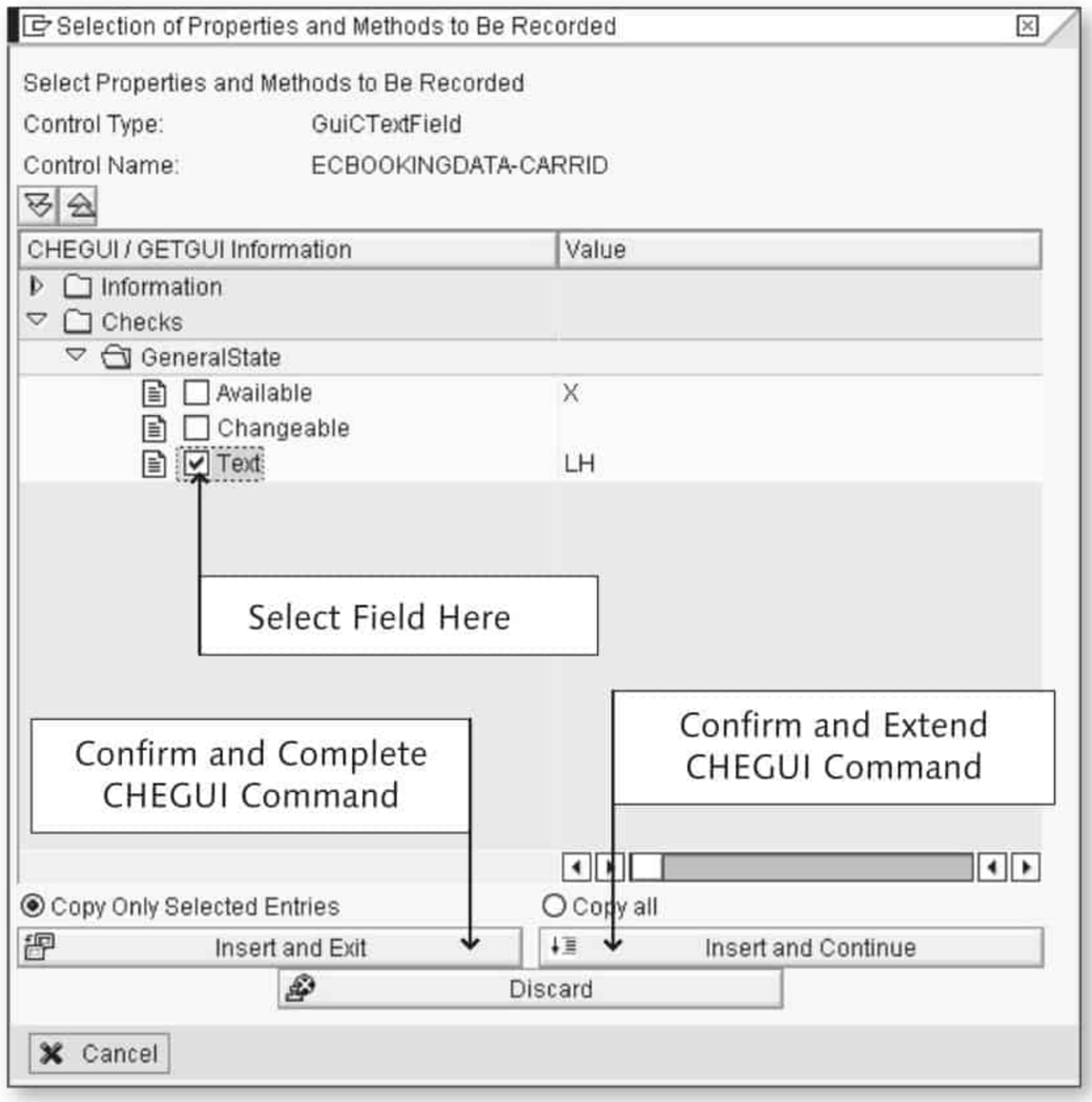


Figure 8.25 Dialog for Processing a "CHEGUI"/"GETGUI" Command

Selecting properties

Because every screen element contains quite a number of properties, you must decide which of these properties you want to check. Usually this is the input value of the field. For text fields, this is the TEXT property to be found under GET/GENERALSTATE/TEXT. For some other screen elements, like list fields, for example, the input value is named VALUE and can be found in the same place.

366 Book_M.indb 441 12/7/10 3:23:16 PM

In some situations it might make sense to check the availability of a screen element before it is accessed. For this purpose, there is an AVAILABLE property. It always has the value "X" at the time of recording. For processing, it only has a value of "X" if the relevant screen element is accessible.

After you've selected the attribute that you want to read or want to read and check, you must select from the following options:

- **Insert and Exit**
  Enters the attributes in the command interface of the `GETGUI/CHEGUI` command and returns to the recording.
- **Insert and Continue**
  Enters the attributes in the command interface of the `GETGUI/CHEGUI` command and returns to the recording. You can insert additional fields in the command interface of the same `GETGUI/CHEGUI` command.
- **Reject**
  You exit the dialog, and the `GETGUI/CHEGUI` command is not inserted into the script.

Initial state recording

Because the eCATT commands `CHEGUI` and `GETGUI` were not introduced until SAP Web AS Release 6.40, a different option of accessing field values has to be used in Web AS Release 6.20. For this purpose, the initial state recording is implemented. For SAP Web AS 6.20, this is the only option to read values. As of SAP Web AS 6.40, it is no longer recommended to use this function.

"Extended" button

Last but not least, the recording window also provides the EXTENDED button. If you select this option, you are provided with the following additional functions (see Figure 8.26):

- In the RECORDING MODE area, you can specify in FOR TREE CONTROLS whether the recording is supposed to be run via a key or path and in FOR CONTEXT MENUS whether the recording is supposed to be implemented based on a key, text, or position.
- In the INSERT COMMENT IN SCRIPT area, you can insert a comment which is inserted into the currently recorded script in the background.

366 Book_M.indb 442 12/7/10 3:23:16 PM

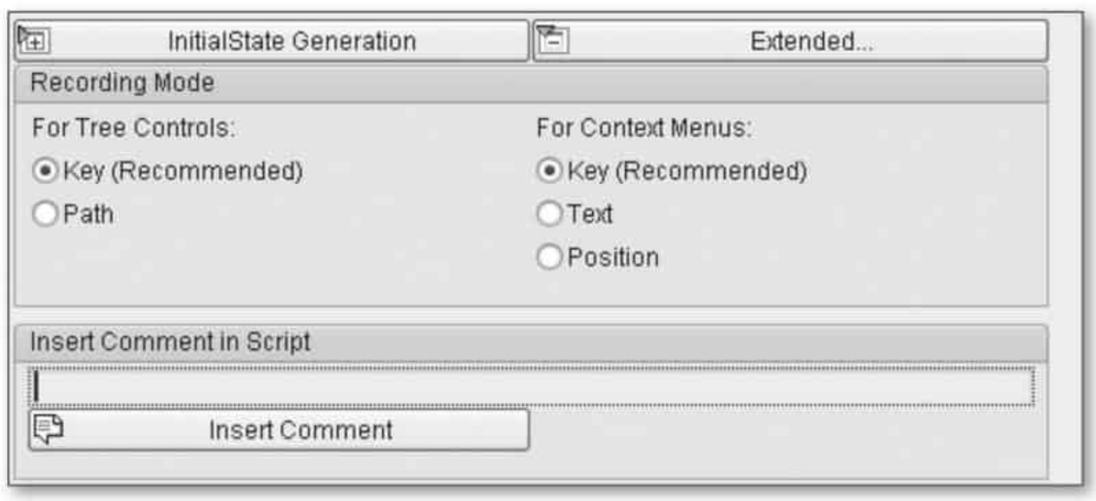


Figure 8.26 Dialog with Expanded Extension Functions

Command interfaces

After recording, you can find all actions performed by the user in the relevant command interface with all details. This interface contains different nodes depending on the command used (`SAPGUI`, `GETGUI`, `CHEGUI`).

"SAPGUI" command interface

The `SAPGUI` command interface comprises the following main nodes (see Figure 8.27):

- The first node below the main folder contains the system information. The ConnectionID and SessionID nodes specify the window in which the system runs the `SAPGUI` command. It is essential that all `SAPGUI` commands have exactly the same values here in order to execute them in the same window. You can directly change the values for connection ID and session ID here. For more details on the IDs, refer to the relevant sections in this chapter.
- Each ProcessedScreen node corresponds to an executed dynpro within the `SAPGUI` command.
- Below the ProcessedScreen node, you can view the UserChanged-State section which lists all actions at GUI element level executed by the user. At this point, you can parameterize the GUI elements; for instance, input fields. For example, the blue arrow pointing to the right just before the term "text" indicates that the user made an entry in an input field. These fields can be parameterized, or more precisely only those dynpro fields can be parameterized that you've changed via user entries during recording. This also means that F4 input helps are not recorded, but only the actually selected value. If in this example we assume that the name "Printer1" was entered as the value, then, a parameter (`I_PRINTERNAME` in this case) can be assigned to this value in the window on the right.

366 Book_M.indb 443 12/7/10 3:23:16 PM

In the USER CHANGEDSTATE section, you can only set the values. With GETGUI and CHEGUI, you can read and check values of output fields.

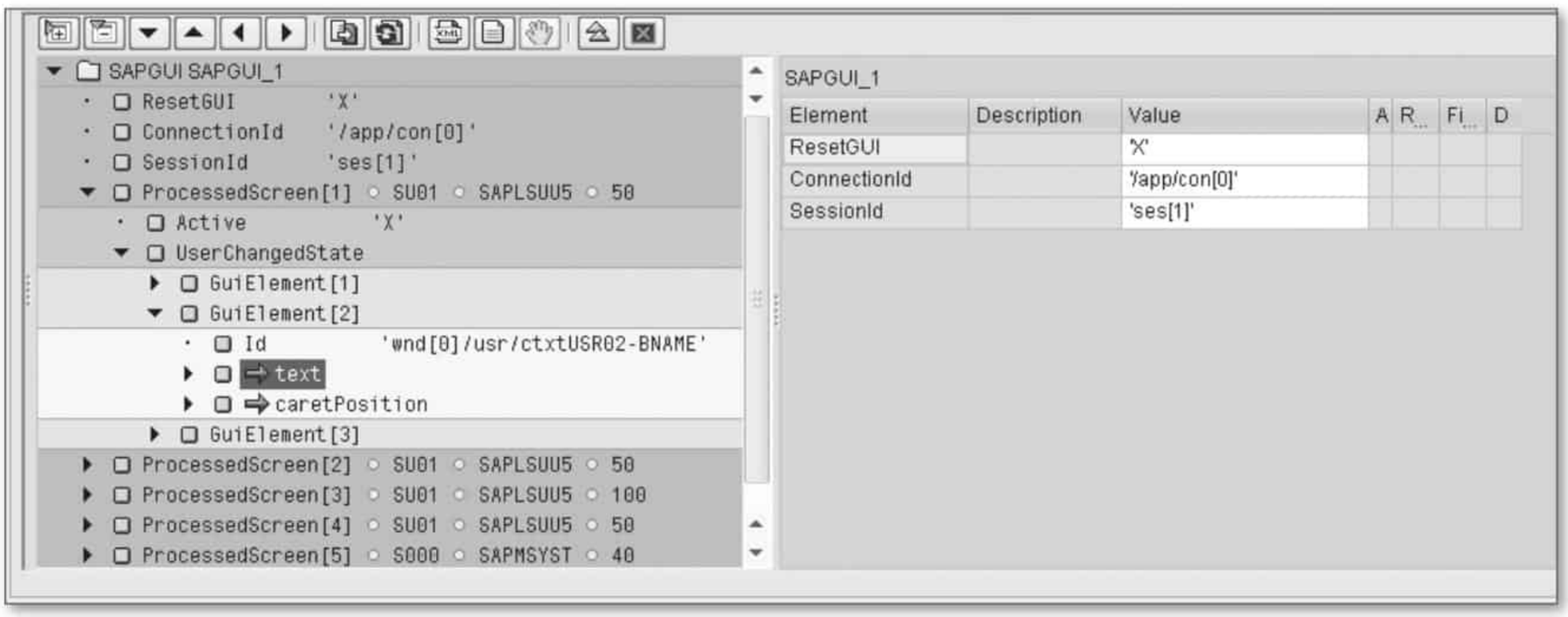


Figure 8.27 "SAPGUI" Command Interface

"GETGUI" command interface

If you inserted a GETGUI command for the recording, you can parameterize the corresponding field in the command interface so that you can exchange the values at runtime.

For this purpose, expand the GUIELEMENT node that contains the selected field, and enter the parameter or variable to be used in the VALUE row (see Figure 8.28).

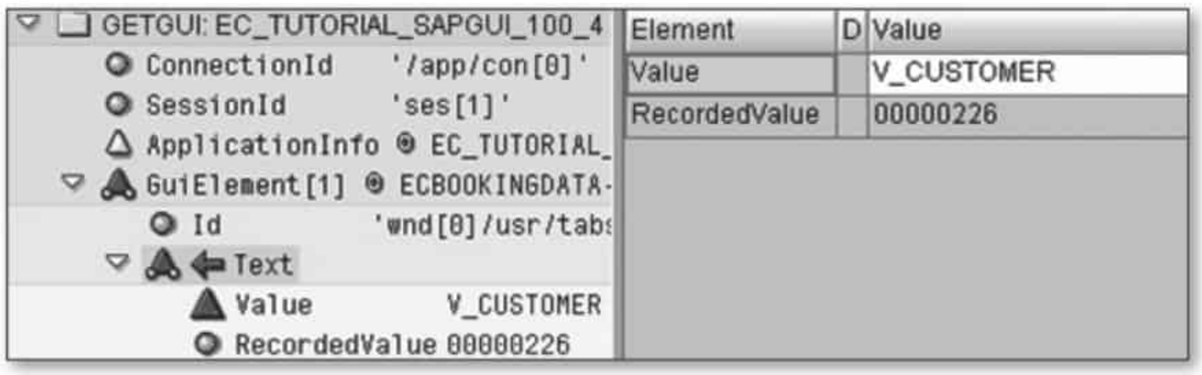


Figure 8.28 "GETGUI" Command Interface

"CHEGUI" command interface

The command interface of CHEGUI is very similar to the GETGUI command interface (see Figure 8.29).

In line with your selection, a CHEGUI command is inserted into your script. To parameterize a field, expand the GUIELEMENT node that contains the selected field. Select a relational operation (=, <>, . . .) in the

366 Book_M.indb 444 12/7/10 3:23:17 PM

CHECKACTION row, and enter the parameter or variable to be used for the check in the EXPECTEDVALUE row. You can also parameterize the VALUE field; there, you find the actual value at the time of execution.

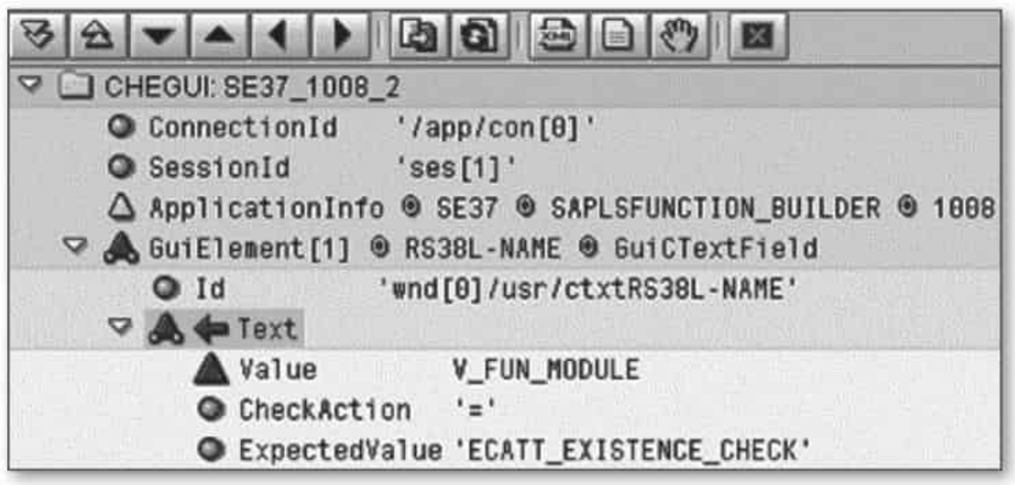


Figure 8.29 "CHEGUI" Command Interface

Revising "SAPGUI" commands

As of SAP Web AS Release 6.40, there are a number of functions for revising SAPGUI commands that have already been recorded. In addition to improving the usability, they primarily simplify the maintenance of the scripts.

SAPGUI attach

An important aspect is that `SAPGUI` commands recorded as of SAP Web AS Release 6.40 can be extended at a later stage. To extend a script, you first need to ensure that you are at that point in the recorded session where you want to insert the extension. Then select PATTERN, and specify UI CONTROL as the group and SAPGUI (ATTACH) as the command. In the RECORD SAP GUI COMMAND WINDOW, define the granularity and whether the command is supposed to be recorded manually or automatically.

After you've activated the ATTACH mode, the system displays a list of all sessions that you can record. This list contains all sessions on the local PC on which scripting is activated in the front end and the back end (except for the eCATT session). Choose the sessions to be recorded; you can activate multiple sessions at a time. Run the script recording as usual. The existing script is extended by adding the new steps.

If you want to use the extension function without working directly in the correct position in the application to be recorded, temporarily comment-out the commands not desired. Then execute the script. Ensure that the DO NOT CLOSE CREATED SESSIONS option has been selected in the script execution dialog. As soon as the session has reached the appropriate state

366 Book_M.indb 445 12/7/10 3:23:18 PM

(in other words, the end of the existing part of the script), return to the script editor. Select SAPGUI (ATTACH) from the pattern dialog.

"Join" and "Split" commands

The function of splitting or joining SAPGUI commands after recording has also been added to the SAP Web AS Release 6.40. This is very useful for inserting CHEGUI/GETGUI commands at a later stage and improving the readability of the script.

Joining "SAPGUI" commands

To join SAPGUI commands, highlight them in the script editor and open the context menu. From the menu, select the JOIN item. A new SAPGUI command is created, and its command interface comprises the actions of all highlighted commands. The commands marked are commented out.

Splitting "SAPGUI" commands

Split functionality divides an SAPGUI command into several commands. This ensures a better overview, but is even more important in a re-recording when you want to split existing commands and insert new steps for re-recording. For example, if you forgot during a recording to fill a text field, you can split the command at the point where the text field needs to be filled. To split a long SAPGUI command, highlight the relevant command in the script editor. Open the context menu, and select the SPLITTING AT submenu. Then select the granularity. The following granularity levels are available:

- Transaction change
- Dynpro change
- Dialog step
- Methods/property

If you split a command in the script editor, the number of new commands depends on the granularity selected.

As of SAP NetWeaver 7.0, you can split in any place. If you split a command in the structure editor, two new commands are created for the resulting parts. Select the SPLIT COMMAND INTERFACE option from the context menu.

The original command is commented out, and the new commands are inserted with the new command interfaces. The original command interface is not lost. You can delete it if you are sure that you no longer require it.

366 Book_M.indb 446 12/7/10 3:23:18 PM

Flexibility of the recording

There are several alternatives for flexibly designing `SAPGUI` commands. This flexibility enables you to cover several test variants with different details in a single test script. For example, you can set the editing of a dynpro to optional. In the command interface of the `SAPGUI` command, set the `ProcessedScreen[n]\Active` value from "X" to "O" for optional. The dynpro is edited only if it is displayed by the application. Otherwise, this part of the script is skipped. The common use of this option is to handle popup windows in the script that, depending on the input data or context, are either displayed or not.

You can create more complex constructs to cover alternative paths using a dynpro. For this purpose, select a fine granularity for the recording (for instance, METHOD) and then toggle between different `SAPGUI` commands using conditionals (see the `IF` command; Section 8.9, *Additional eCATT Commands*).

Start options for the SAPGUI driver

When processing a script using `SAPGUI` commands, you are provided with separate SAPGUI-specific start options (see Figure 8.30).

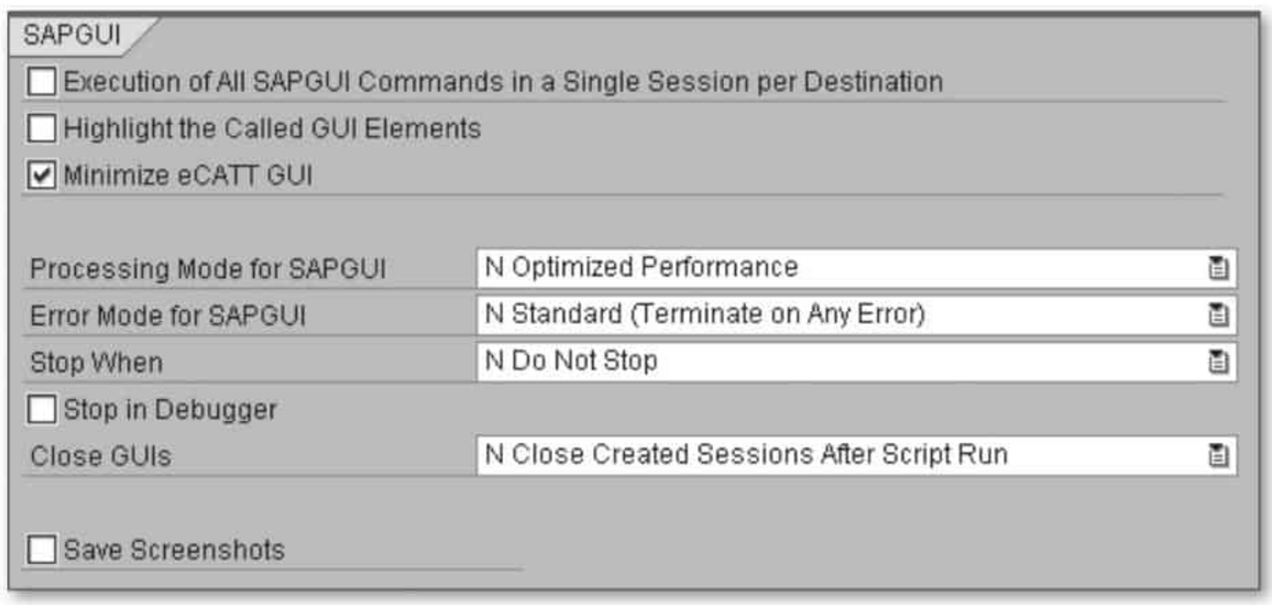


Figure 8.30 Start Options for Executing "SAPGUI" Commands

- The option EXECUTION OF ALL SAPGUI COMMANDS IN A SINGLE SESSION PER DESTINATION causes one session to be used per destination (target system). This is useful if you have different combinations of connection ID and session ID in your script or if you encounter difficulties with opening a new session due to a stored limitation of the number of sessions (default: 6).
- If the HIGHLIGHT THE CALLED GUI ELEMENTS option is enabled, the active screen element is highlighted with a red frame while the script

366 Book_M.indb 447 12/7/10 3:23:18 PM

is being executed. This option can be very useful for debugging scripts.

- The MINIMIZE eCATT GUI option minimizes the SAP GUI window running eCATT to an icon on the task bar.
- The PROCESSING MODE FOR SAPGUI option is a performance parameter. The following modes can be selected:
  - **Optimized Performance**
    In this mode, the GUI updates are processed via the *automation queue*, which forwards them to the GUI. On the one hand, this improves the performance. On the other hand, a possible error in the script may not be displayed in the eCATT log directly where it occurs. Nevertheless, this is the recommended execution mode.
  - **Synchronous GUI Control**
    Bypasses the automation queue and sends GUI updates directly to the front end. In this mode, GUI updates are sent to the GUI synchronously.
- The ERROR MODE FOR SAPGUI option controls the behavior of eCATT in the case of an error. The selections are self-explanatory.
- The STOP WHEN option causes eCATT to interrupt the script execution in specific places. It is not continued until the user confirms it. This option is particularly useful during script development. In case of an interruption, the execution control operates via a popup window.
- If you additionally enable STOP IN DEBUGGER, the eCATT script switches to the debug mode whenever it is interrupted. In this case, the execution control is via the debugger and not via a popup window. More information can be found in Section 8.12, *Further Steps*.
- The CLOSE GUIs option enables you to specify whether and when the modes that have been automatically created during the script execution are automatically closed again.

Screenshot functionality

As of SAP NetWeaver Release 7.00, functionality for automatically creating and saving screenshots is provided (see Figure 8.31). This functionality was designed primarily for covering the documentation requirements in an environment where validation is mandatory. However, a sequence of screenshots can also be useful for tracing the individual steps of test runs for troubleshooting purposes.

366 Book_M.indb 448 12/7/10 3:23:18 PM

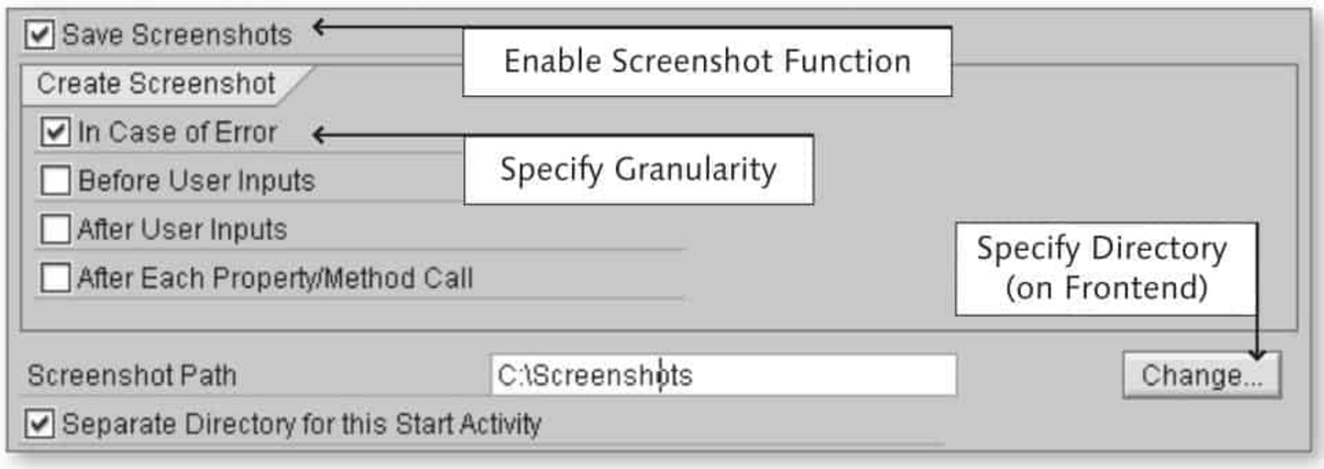


Figure 8.31 Screenshot Options in the Start Options

To enable the functions for automatically creating screenshots, select the SAVE SCREENSHOTS option in the start options. The screenshot options are displayed. Specify the granularity and a directory where the screenshots are to be stored. The specification of the granularity level defines the application events for which the screenshots are to be created. The screenshots are saved in .jpg format.

Because the functionality for automatically creating screenshots requires the availability of a user interface, it is only available for the SAPGUI driver. In an environment requiring validation, you should consider this when you select the test driver. It may make more sense in such a case to use the SAPGUI driver even for recording a transaction without controls, even though you would normally use the TCD driver due to its better maintainability and performance.

Regarding the message handling within the SAPGUI driver, please refer to Section 8.6.3, *Message Handling*.

8.2.3 Testing Web Dynpro Applications

As of SAP Web AS 6.40, eCATT supports the direct testing of Web Dynpro Java-based applications. For SAP NetWeaver 7.02 and as of SAP NetWeaver 7.20, the testing of Web Dynpro ABAP-based applications is also supported.

When recording a Web Dynpro application, the user operates it either in a Web Dynpro client (for Web Dynpro ABAP and Web Dynpro Java) or via the browser (only for Web Dynpro Java). Figure 8.32 shows the recording via the Web Dynpro Business Client.

Architecture

366 Book_M.indb 449 12/7/10 3:23:23 PM

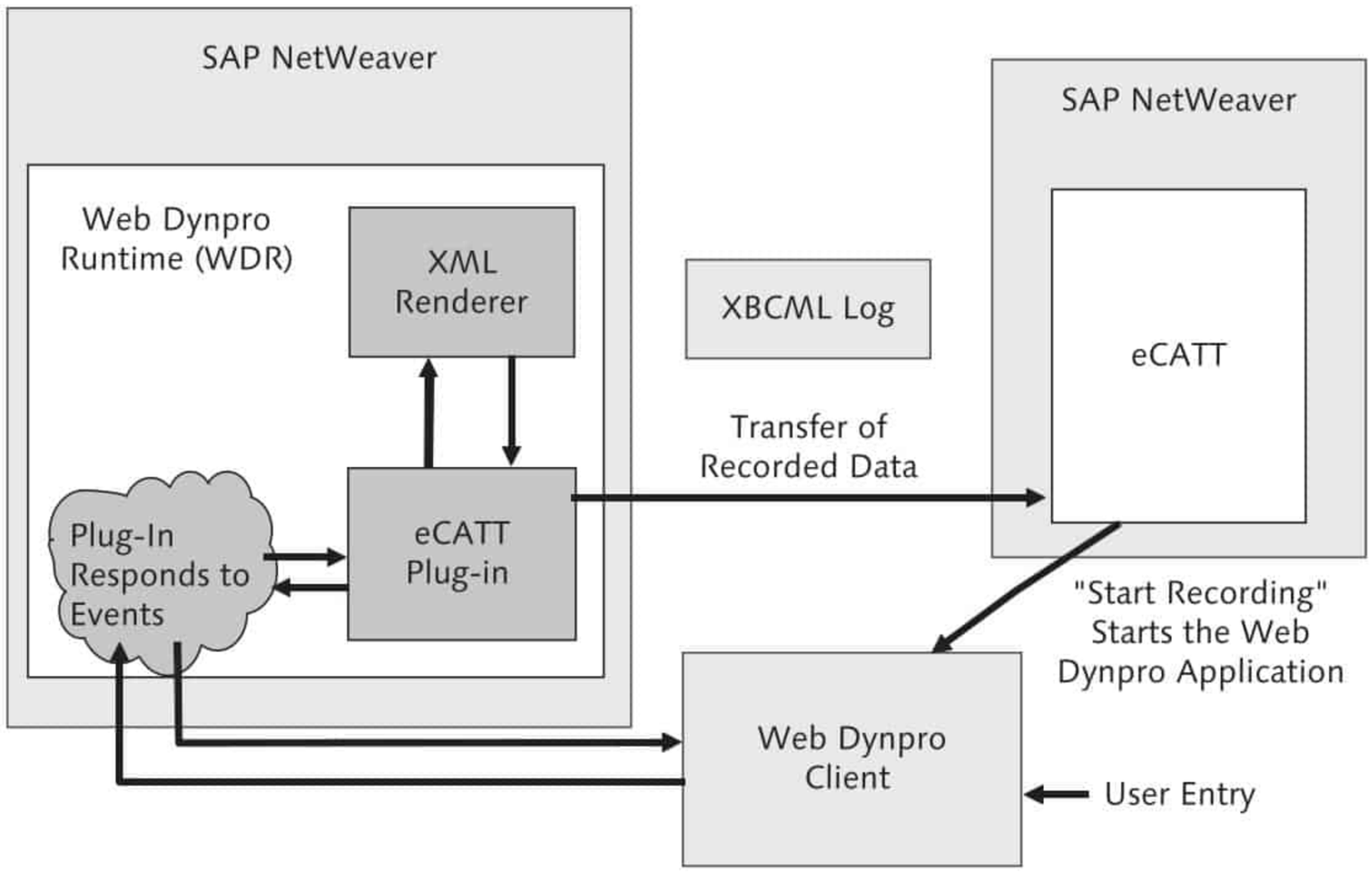


Figure 8.32 Recording Web Dynpro Commands

Communication while recording

When you start the recording of a Web Dynpro in Web Dynpro ABAP, the Web Dynpro Client opens. In Web Dynpro Java, the user can select between the recording in the Web Dynpro Business Client and recording via the browser. The user operates the application in the Web Dynpro Client or in the browser. As soon as changes are made to the Web Dynpro context (back end) that is, if data change or actions are executed the plug-in is informed about this and then records the operations. In Web Dynpro, eCATT records the business logic of the application and not the events in the interface as in the SAP GUI with controls, for example. The data recorded is then sent to eCATT as an XML file in the background.

Communication while processing

The processing of Web Dynpro applications is done in the background via HTTP without a browser.

During processing, the request is sent to the target system via HTTP in XML format. The request corresponds to an input in the UI plus the Web Dynpro action. An XML description of the next page to be displayed is returned as the response. The description contains the structure of the window or the subareas of a window as well as the data. If, for example, an input field is filled with values, this value is also part of the client's response. The processing usually occurs in the background. Using the simulator, you can trace the processing of the transactions on screen.

366 Book_M.indb 450 12/7/10 3:23:23 PM

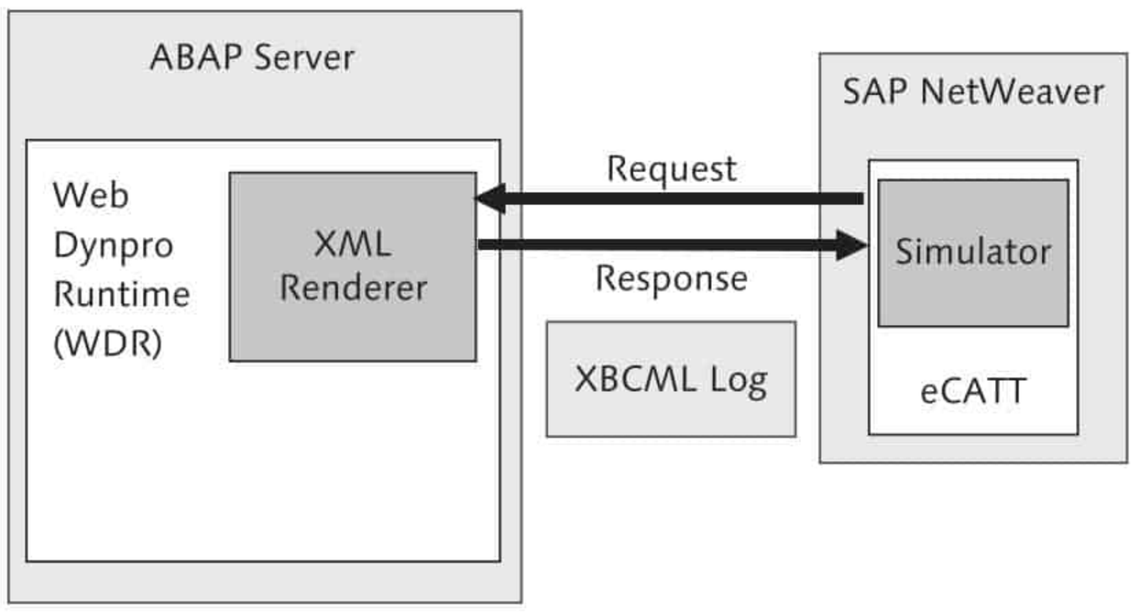


Figure 8.33 Processing a Web Dynpro Transaction

Structure of the URL

The URL for addressing the Web Dynpro application is structured as follows:

```
HTTP://<Server>:<Service>/<URL extension>/<Application>
```

The meanings of server (name or IP address) and service (port) are familiar. The URL extension looks different depending on whether the Web Dynpro runtime environment is based on the Java or the ABAP stack. If you are using the ABAP stack, the extension is:

```
<ABAP URL extension> = sap/bc/webdynpro
```

If the Web Dynpro runtime environment is based on Java, the extension has the following format:

```
<Java URL extension> = webdynpro/dispatcher
```

Creating the connections

The first step for recording a Web Dynpro-based application is the creation of the targets for the HTTP connections. Use Transaction SM59 for this purpose. Java-based applications require an HTTP connection to an external server (connection type G). For ABAP applications, you should create an HTTP connection to the SAP system (connection type H).

Enter host name and service (port) of the target system as shown in Figure 8.34.

As soon as you have created and successfully tested the HTTP connection, insert a logical target in the system data container. In the HTTP DESTINATION column, store the newly created connection for this target system.

366 Book_M.indb 451 12/7/10 3:23:23 PM

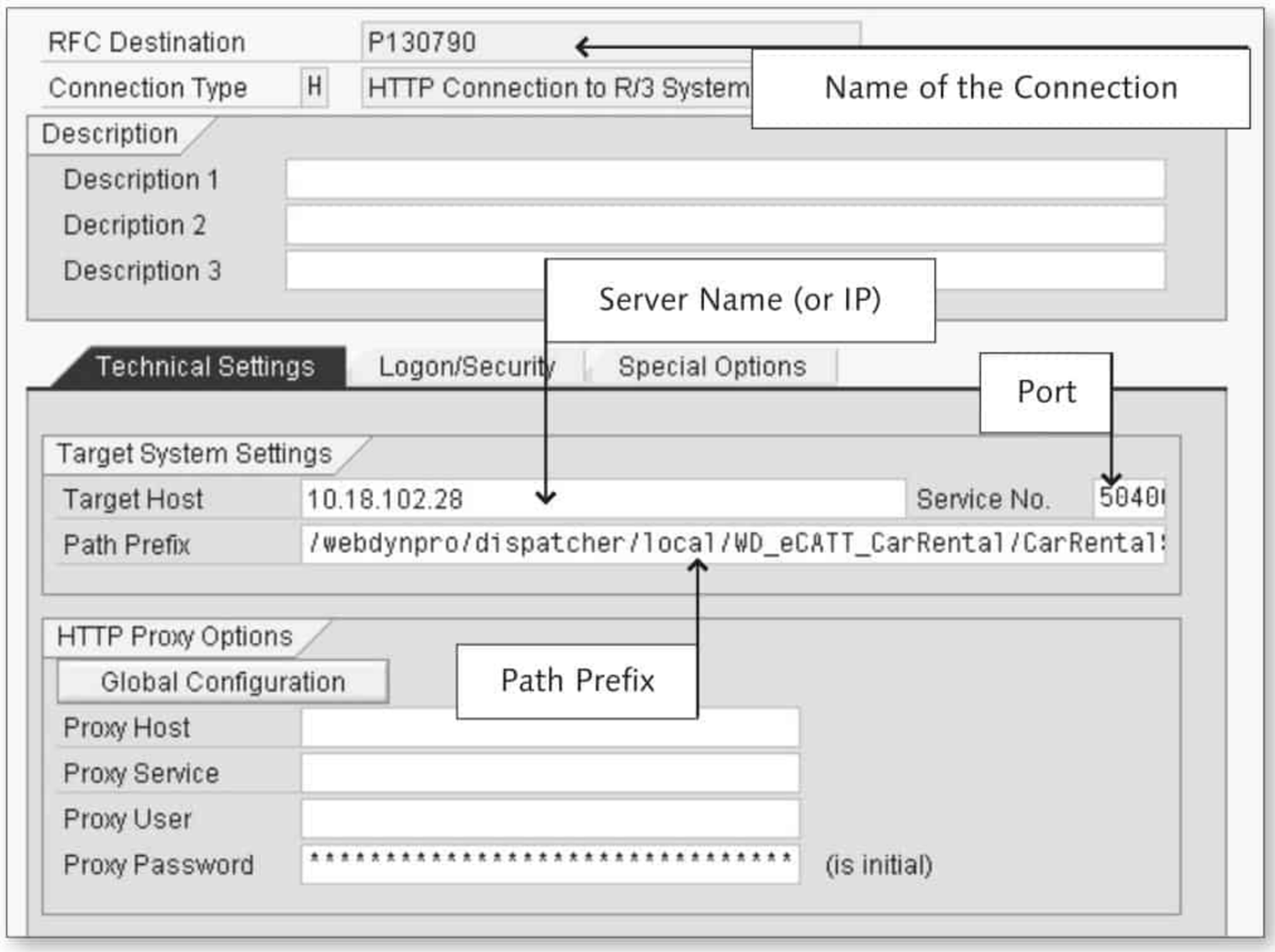


Figure 8.34 Creating a Connection Transaction SM59

Recording Web Dynpro transactions

To record a WEBDYNPRO command, go to the script editor. Select PATTERN, and from the UI CONTROL group select the WEBDYNPRO command. A dialog box for specifying the Web Dynpro application is displayed. This dialog box differs, depending on whether you want to record Web Dynpro ABAP or Java. In case of Web Dynpro Java, you have the option of starting the recording in the Web Dynpro Client or in the browser. In case of Web Dynpro ABAP, only the recording button is available, and the recording is started automatically in the Web Dynpro Business Client when you select this button. Select a target system (the target system with the HTTP connection you previously created; see Figure 8.35).

In the APPLICATION input field, complete the basis URL of the connection with an application-specific section. This section depends on the application to be recorded; for example, CarRental. The address sections are then merged by eCATT. Then, in the case of Web Dynpro Java, select the recording type (in the browser or Web Dynpro Business Client). Subsequently, click the START RECORDING button. Note that the Web Dynpro Business Client must be installed to be able to start the recording in the Business Client (see SAP Note 773899).

366 Book_M.indb 452 12/7/10 3:23:26 PM

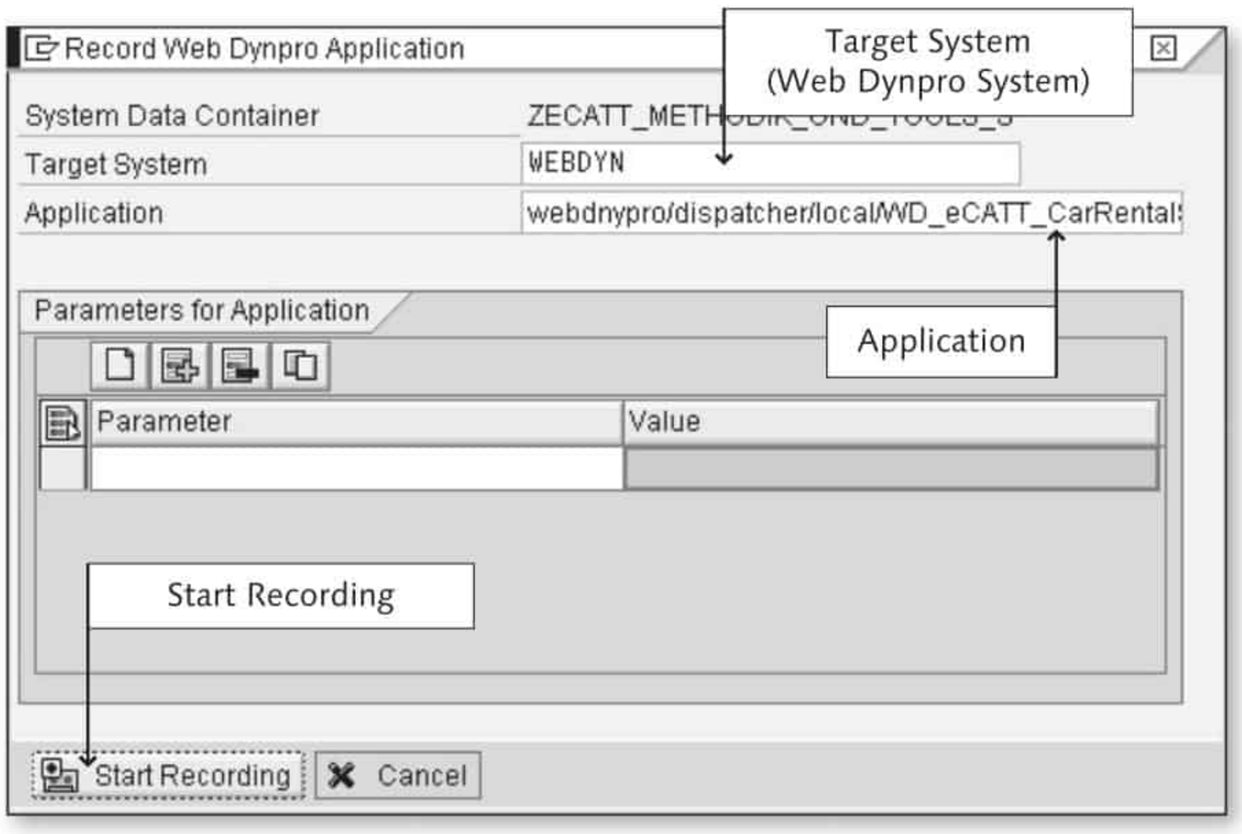


Figure 8.35 Starting the Web Dynpro Java Recording

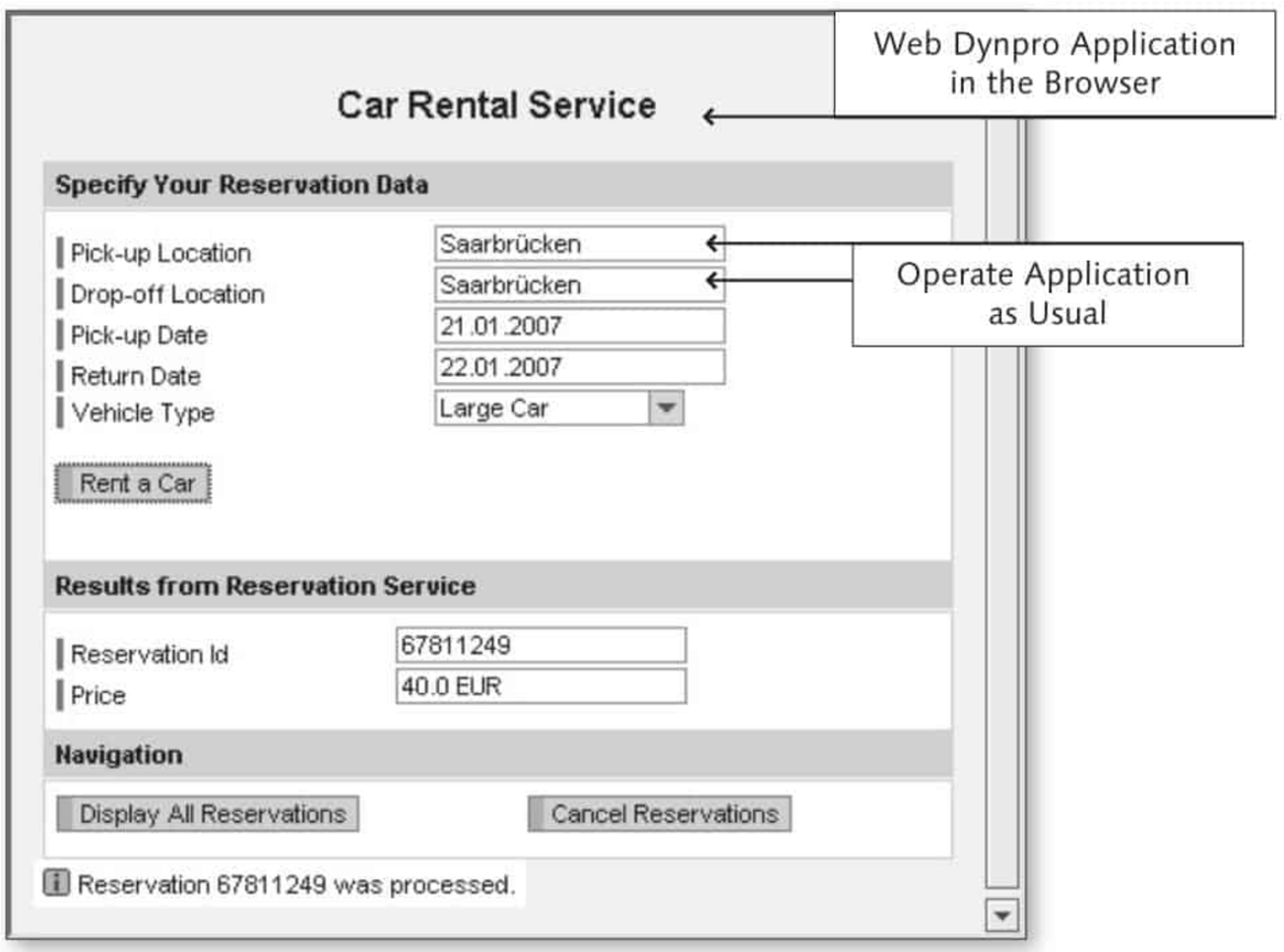


Figure 8.36 Browser Window with Web Dynpro Application During Recording

As soon as you start recording, eCATT automatically opens a browser window or Web Dynpro Business Client window containing the Web Dynpro application (see Figure 8.36). In this window, you operate the application as usual and populate the dialog elements with values. Your

366 Book_M.indb 453 12/7/10 3:23:36 PM

input is recorded and is later available in the command interface of the WEBDYNPRO command.

At the same time, another window opens in which you can stop recording again (see Figure 8.37).

Figure 8.37 Dialog Window for Stopping the Recording

As soon as you have stopped the interaction with the Web Dynpro application, use the task bar to change to this window and stop the recording. Now the WEBDYNPRO statement is displayed in the script editor.

"WEBDYNPRO" Command

The WEBDYNPRO command is designed for addressing the Web Dynpro driver. It has the following notation:

```
WEBDYNRPO (<Command interface>, [<Target system>] ).
```

The target system must be an HTTP connection of the G type (for Java Server) or the H type (for ABAP Server).

Command interface

The command interface of the WEBDYNPRO command comprises the following nodes with recording details (see Figure 8.38):

- Under SCREEN • DATA, you can find the XML description of the page returned. This description is relevant if the pages are supposed to be displayed in the eCATT Web Dynpro simulator.
- Under DATACHANGES, you can find the values entered during the recording. Like with an SAPGUI command, you can parameterize the recording to link the Web Dynpro command to the test data.
- Under ACTION, you can find the actions that trigger a round trip. For example, data changes do not become effective until the relevant action triggers the round trip.
- Under GETS AND CHECKS, the system displays the details on checks or value determinations. This node is only visible if such checks/value determinations have been started from the simulator (see next section).

366 Book_M.indb 454 12/7/10 3:23:36 PM

- Under PAGE • SCREEN • MESSAGES, you can find the messages sent during recording. You can process messages from Web Dynpro applications via a `MESSAGE ... ENDMESSAGE` block as well (see Section 8.6.3, *Message Handling*).

Figure 8.38 Command Interface of the Web Dynpro Command

Simulator

The `WEBDYNPRO` command includes a page simulator which enables you to determine values from output fields or execute checks on their content. To start the simulator, open a `WEBDYNPRO` command interface in the structure editor, and select the magnifying glass button. In the simulation, select the field to be checked, and then choose INSERT CHECK. If you want to determine the value of a field, select INSERT GET instead. In both cases, a new node, GETS_AND_CHECKS, is inserted in the command interface. In this node, you can now determine or test the values of a page. The parameterization is similar to that from the `GETGUI` and `CHEGUI` commands.

You can also directly go to the simulator display from the log if you double-click the PAGE node. This way, you can immediately determine to which screen a possible error refers. Even in the debugger, you go to the simulation if you press the F5 key. Note that the SAP NetWeaver Business Client must be installed to use the eCATT Web Dynpro simulator. SAP Note 773899 provides more detailed information.

Start options for the Web Dynpro driver

When the Web Dynpro driver is used, you have the following start options:

- If you select the PROCESS IN BACKGROUND option, the script is processed without being displayed in a user interface. The progress of the script cannot be traced on screen. However, this option ensures the best performance.

366 Book_M.indb 455 12/7/10 3:23:37 PM

- If you select the PROCESS IN FOREGROUND option, the progress of the script is displayed in a user interface. The PROCESS IN FOREGROUND (DISPLAY RECORDED PAGE IN PARALLEL) option causes the progress of the test case in the application at the time of recording to be displayed in a second window in addition to the actual script. Because it enables comparison, this option is very helpful for the troubleshooting process during the script development. For both options for processing in the foreground, you can specify how long the respective screen is supposed to be displayed.

SE80 connection

You can also start the eCATT recording of a Web Dynpro application directly from the object navigator (Transaction SE80; see Figure 8.39). Here, the benefit is that you don't need to specifically call the eCATT Transaction SECATT. An XML file is returned.

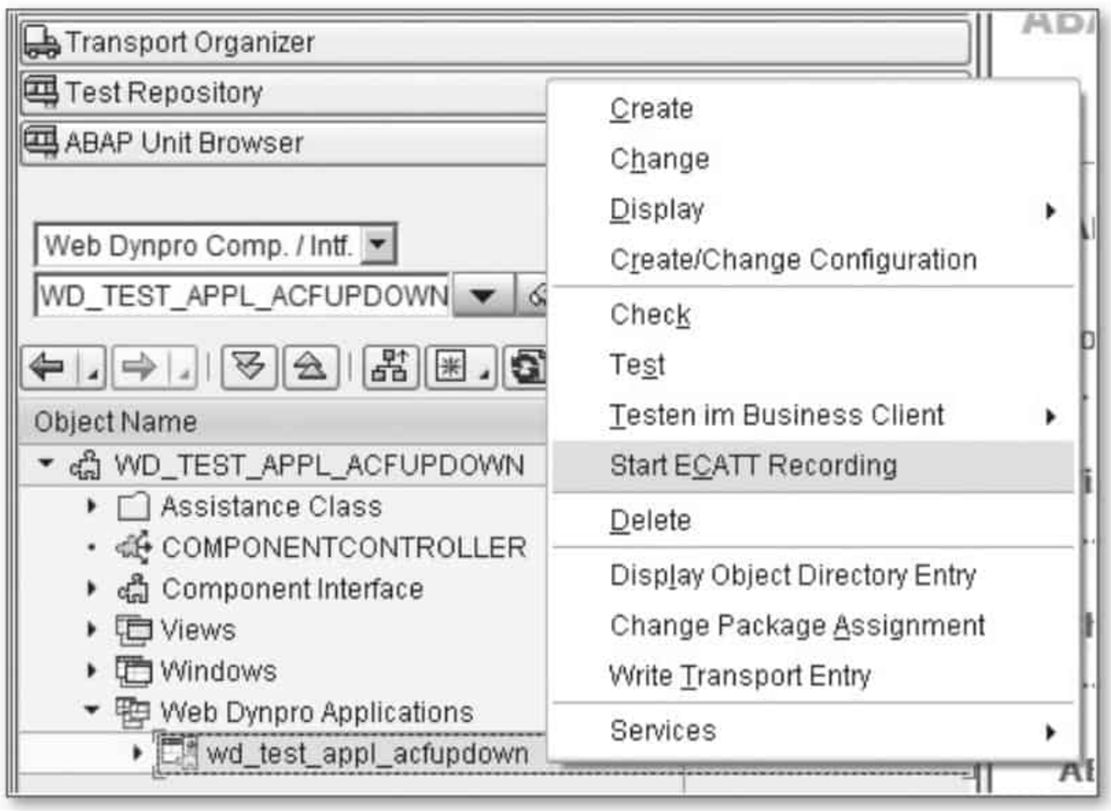


Figure 8.39 Starting the Recording from Transaction SE80

If you select the Pattern button in Transaction SECATT, two additional Web Dynpro commands are available for selection within the UI CONTROL group: `WEBDYNPRO (attach)` and `WEBDYNPRO (import)`. The latter option refers to the Web Dynpro Java applications. If a Web Dynpro Java application is recorded in SAP NetWeaver Developer Studio, the system generates an XML file that can be uploaded to eCATT. Then, the corresponding Web Dynpro commands are generated in eCATT. `WEBDYNPRO (attach)` enables you to extend scripts from the Web Dynpro simulator.

366 Book_M.indb 456 12/7/10 3:23:37 PM

8.2.4 Summary

UI-based applications can be tested with eCATT via different test drivers.

Transactions without controls can be recorded by the TCD driver: comparable to a macro recorder that you might know from Microsoft Excel. During parameterization, you replace the fixed values entered during the recording process with parameters and thus can create a flexible, usable script. Changing the assignment mode enables you to read and check the actual values of a parameter.

Transactions with controls can be recorded using the SAPGUI driver. The selection of the correct granularity is very important, although it can be corrected at a later stage in more recent eCATT versions. For reading and checking values, the `GETGUI` and `CHEGUI` commands are available. Moreover, you can join and split script commands using the `JOIN` and `SPLIT` commands.

Web Dynpro applications can be recorded just like SAP GUI transactions. The URL is opened automatically in the Web Dynpro Business Client or a browser. The recording takes place in parallel while the Web Dynpro client or the browser is being operated by the SAP application.

8.3 Creating Tests Via Direct Program Control

eCATT enables you to test the following program controls:

- Global ABAP object classes
- Function modules and BAPIs
- Inline ABAP
- Database accesses

8.3.1 Testing Global ABAP Object Classes

The eCATT command set comprises commands for testing global classes. You can find the set of commands if you open the desired test script in transaction SECATT, click the PATTERN button, and select the ABAP OBJECTS group.

To be able to access instance attributes and methods of a class, you must create an instance of the class. As a prerequisite, there must be a param-

366 Book_M.indb 457 12/7/10 3:23:37 PM

eter of the type to which the object can be assigned. The parameter must contain the type of the class and a class or an interface from which the class inherits.

By default, an object is created using the `CREATEOBJ` command. However, you also can create an object by using a function module or a method that provides an object.

Instance attributes and static attributes

To access instance attributes, you first need to instantiate the object. The following table lists commands available for accessing instance attributes and static attributes of a class.

Command For Accessing Instance Attributes	Command For Accessing Static Attributes	Description
`CHEATTR`	`CHESTATIC`	Compares the current value of an attribute with the value specified.
`GETATTR`	`GETSTATIC`	Calls the current value of an attribute and assigns a parameter to this value.
`SETATTR`	`SETSTATIC`	Changes the current value of an attribute to a value specified.

Table 8.3 Commands for Accessing Instance and Static Attributes

Methods

You use `CALLMETHOD` to call an instance method of a class. For this purpose, you must first instantiate the object. To call a static method, use `CALLSTATIC`.

8.3.2 Function Modules and BAPIs

In the following cases, it could be necessary to test or use function modules:

- In a test in which you test an individual function module.
- In a background process in which you test a chain of function modules which represent a complete process by transferring results of a module to the next one. You can perform this work step in a single script or in several consecutive test scripts.
- As a utility within a script; for instance, to call data from the database which you want to use in transactions or to perform complex plausibility checks via already existing function modules.

366 Book_M.indb 458 12/7/10 3:23:37 PM

"FUN" Command

FUN

The eCATT command required to call function modules is named `FUN`. After opening a test script in Transaction SECATT, you can select the PROGRAM CONTROL group via the Pattern button.

The command interface corresponds to the interface of the function module. In the structure editor, it is subdivided into areas for the different parameter types—import, export, change, and tables—with an additional area for exceptions. For the transfer to a function module, you assign the relevant literal or variable to the correct parameter in the command interface. To call a value from an export parameter of the function module, you assign a variable to the parameter.

8.3.3 Inline ABAP

Inline ABAP provides the option of inserting complex programming elements into an eCATT script. An inline ABAP block is initiated with the `ABAP` script statement and ends with `ENDABAP`. This statement can also be selected via the Pattern button in the PROGRAM CONTROL group.

Within the inline ABAP block, you can use (almost) any ABAP commands. Inline ABAP is particularly required for complex database accesses, but it can also be used for all calls of function modules and ABAP OO routines.

Note

When you use inline ABAP, you should avoid any user interactions as well as mode-generating or mode-ending operations (for instance, `CALL FUNCTION STARTING NEW TASK`) because otherwise an automatic test execution is no longer possible.

The connection to the eCATT script is established via the local variables (parameter type V). All local variables are automatically declared within inline ABAP and are provided with the current values from the script execution. After executing the inline ABAP block, all local variables are in turn provided with the values that were assigned within the inline ABAP.

Tip

You should outsource inline ABAP scripts to custom scripts in order to keep the number of local variables as low as possible. Transfer to and from inline ABAP can hurt performance when complex variables and large tables are used.

366 Book_M.indb 459 12/7/10 3:23:38 PM

8.3.4 Database Accesses

CHETAB

"CHETAB" Command

The CHETAB command compares field values of a database table with values which you define in the command interface; the command interface has the same structure as the table.

Using the CHETAB command, you can check the following:

- Database updates after transactions
- The existence of specific table entries
- Entries in customizing tables

You maintain the values of fields in the command interface. For this purpose, you aren't limited to the key fields, and you don't need to specify the complete key either. However, you shouldn't leave any key field empty; we recommend entering an asterisk in each key field to which you don't assign a value.

If no data record is found that meets the search criteria, an error message is written to the log. If one or more data records are found, the check is entered as successful in the log.

You can specify the system in which the database table can be found. If you specify a target system, a corresponding system container must be assigned to the test script or the test configuration.

GETTAB

"GETTAB" Command

The GETTAB command reads a data record of a database table and assigns the values of the data record to the corresponding fields of a command interface. The command interface has the same structure as the table.

In general, the same conditions as apply to CHETAB apply to the maintenance of field values. For this reason, if you don't specify the full key, multiple data records can meet the selection criteria. GETTAB reads the first data record that meets the selection criteria.

366 Book_M.indb 460 12/7/10 3:23:38 PM

8.3.5 Summary

Besides UI-based applications, you can use eCATT to test the following program-controlled applications: ABAP object classes, function modules and BAPIs, and inline ABAP, as well as database accesses.

8.4 Creating Tests for Web Services

The SAP NetWeaver Application Server enables enterprises to extend their solutions by integrating web services and providing them to their users. It supports the XML, SOAP (Stateless, Stateful, and Security), WSDL, and UDDI standards.

As of SAP NetWeaver 7.0, eCATT supports the automated testing of web services. A web service is then called in the same way as an internal ABAP function module. The necessary ABAP proxy classes are generated automatically.

Testing third-party solutions

Because the functionality of the eCATT web service driver is not limited to testing web services provided via the SAP NetWeaver Application Server, you indirectly gain an interesting alternative for testing third-party solutions. As long as the third-party system in your process chain has a web service interface, you can integrate it seamlessly in the test coverage via eCATT scripts.

To test a web service, first generate an HTTP connection using Transaction SM59. Then, in the eCATT script editor, click on the PATTERN button and from the ENTERPRISE SERVICES group select the WEBSERVICE command. Figure 8.40 shows the dialog field that is displayed.

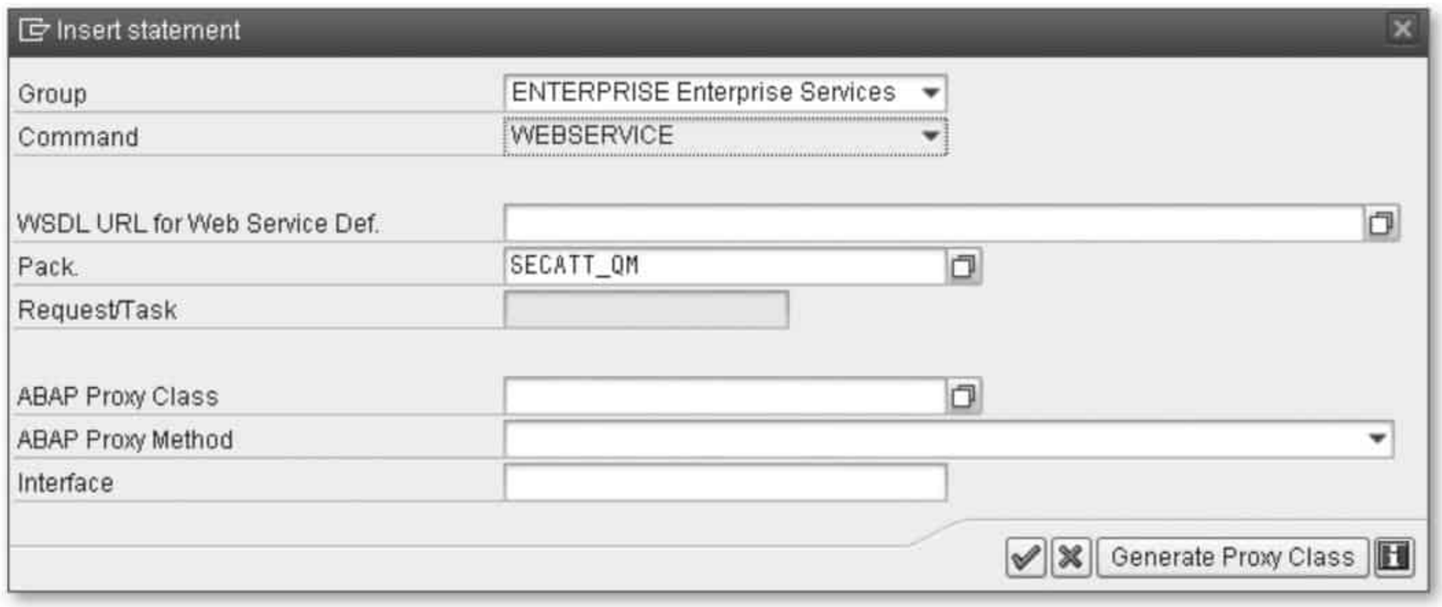


Figure 8.40 Inserting a Web Service Test

366 Book_M.indb 461 12/7/10 3:23:38 PM

Because web services are function-like objects that do not allow for direct user communication, no recording occurs in this step. Instead, you specify a method call that is submitted to the web service.

ABAP proxy objects and WSDL

If the web service is implemented on an SAP NetWeaver Application Server and you know the ABAP proxy class on which it is based, you can select the corresponding ABAP proxy class from the directory. Once you've confirmed your entries, the appropriate values are inserted in the fields, ABAP Proxy Method and Interface.

If there is no appropriate ABAP proxy class yet, eCATT supports you by automatically creating an appropriate ABAP proxy class. In the WSDL URL for Web Service Definition field, enter a URL where a web service description can be found. The WSDL description (Web Service Description Language) specifies the functions provided by the service. Usually, you will want to obtain the description directly from the used server. In that case, the URL normally uses the following format:

```
HTTP://<Server>:<Port>/<Web Service Name>/Config?wsdl
```

As soon as the correct address for the service description has been entered, click on the Generate Proxy Class button; eCATT queries the capabilities of the web service and generates an ABAP proxy class. This generated class is entered directly in the ABAP Proxy Class field. You need to assign the class to a package.

Once you have a functioning proxy class, you can select the visible methods of the class in the ABAP Proxy Method field. Select the wanted method from the list, and close the dialog box. eCATT then generates a `WEB SERVICE` command in the script editor.

"WEB SERVICE" Command

The `WEB SERVICE` command is used for addressing web services. It has the following format:

```
WEB SERVICE(<Command interface>, [<Target system>] ).
```

The command interface (see Figure 8.41) corresponds to the interface of the selected function from the ABAP proxy class. Usually, it is generated automatically from the WSDL. Therefore you do not need to worry about the structure of the transferred parameters. Just populate the command interface with appropriate values.

366 Book_M.indb 462 12/7/10 3:23:38 PM

Figure 8.41 Command Interface of the "WEB SERVICE" Command

Uploading web service test scenarios

With SAP NetWeaver 7.1, you can upload web service test scenarios as XML files into eCATT to test `WEB SERVICE` commands with eCATT. For this purpose, you need the Composition Environment as of Release 7.1. Moreover, the Web Service Navigator must be configured accordingly. Prior to the uploading process, you can make presettings for parameterization. You can select which import or export parameters should be created.

The following upload options are available:

- Upload of the test scenario and simultaneous generation of eCATT objects (test configuration and test script). For this purpose, open Transaction SECATT, enter the name of the test script you want to create or overwrite, and select eCATT OBJECT • OTHER FEATURES • UPLOAD WEB SERVICE TEST SCENARIO DATA.
- Upload of the test scenario and simultaneous generation of a new web service command interface. For this purpose, open Transaction SECATT of the wanted test script, and select the PATTERN button. The INSERT PATTERN selection screen is displayed. In this screen, select ENTERPRISE SERVICES as the group and WEB SERVICE (UPLOAD) as the command.
- Upload of the test scenario in an already existing command interface. For this purpose, open in transaction SECATT the wanted test script, and double-click the web service command interface that is supposed to be uploaded to the file. The command structure is displayed in the subscreen on the right. Then select the UPLOAD WS TEST SCENARIO option.
- Upload of a test scenario to create test configuration variant(s). For this purpose, open the test script in Transaction SECATT and double-click the web service command interface that is supposed to be uploaded

366 Book_M.indb 463 12/7/10 3:23:39 PM

to the file. The command structure is displayed in the subscreen on the right. Then select the CREATE CONFIGURATION VARIANT option.

Except for the option to upload to an existing command interface, all upload variants are "capable of mass processing;" that is, multiple data records can be selected for processing.

After the upload process, the script editor displays a log including all originally selected settings and possible error messages. This enables you to reproduce the upload process.

Integration of Web Dynpros and web services

An interesting application scenario is created when web services are tested that are integrated with Web Dynpro-based transactions. For example, a data record can be entered via an automated Web Dynpro transaction to check the correct update of the data in the test system via an appropriate web-service call. This enables a continuous test of service-oriented solution landscapes.

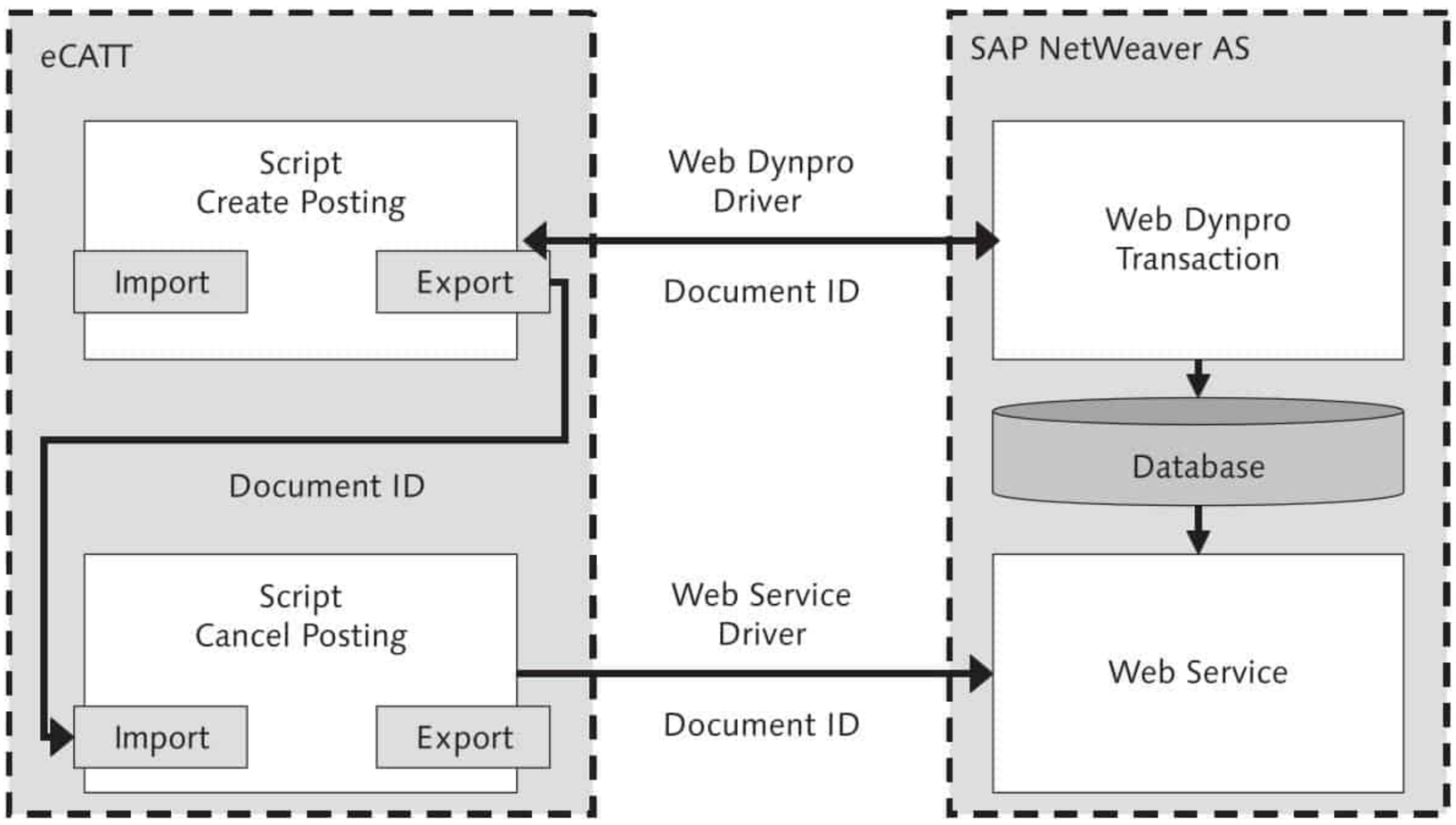


Figure 8.42 Example of a Test Scenario with Web Dynpros and Web Services

Conclusion

The web-services test is fully supported and integrated with eCATT as of SAP NetWeaver 7.0. Because ABAP proxy classes are generated

366 Book_M.indb 464 12/7/10 3:23:39 PM

automatically from WSDL descriptions, addressing a web service is just as simple and comfortable as calling a function module. In combination with Web Dynpros, this results in very elegant possibilities for testing modern service-oriented solution landscapes via different forms of access.

8.5 Integration with External Test Tools

The driver for external test tools plays a special role among the eCATT test drivers. An external tool is a program of a third-party provider that uses the implementation of the BC-eCATT interface to interact with eCATT. SAP can certify the successful interaction. Under *http://www.sap.com/ecosystem/customers/directories/SearchSolution.epx, you can find a list of the test tools certified by SAP for the* interaction with eCATT. Then select BC-ECATT 6.2WIN in the first dropdown list.

In heterogeneous solution landscapes, business processes can be effectively tested automatically with eCATT. The external test tool then covers those process steps that do not use a SAP GUI for Windows or Web Dynpro as a user interface.

BC-eCATT

Such an external test tool must be installed on the front end to be tested, and registered in the back end. The documentation of the external test tool contains information on the registration. In registration, the external program works as an adapter for the software to be tested. The work process—that is, the recording, editing, and processing—of user interactions is controlled via the external tool (see Figure 8.43). For this purpose, the tool is addressed by eCATT via the BC-eCATT interface. Recorded scripts are transferred via BC-eCATT and stored and managed in the central repository like native eCATT scripts. This ensures central, continuous, and consistent data storage even for the integration of external tools.

Recording

For recording, a distinction is made between two different processes. For the integrated recording scenario, eCATT is the driver that calls the external tool. In the standalone scenario, however, the recording is initially run in the external tool and then transferred to eCATT.

366 Book_M.indb 465 12/7/10 3:23:39 PM

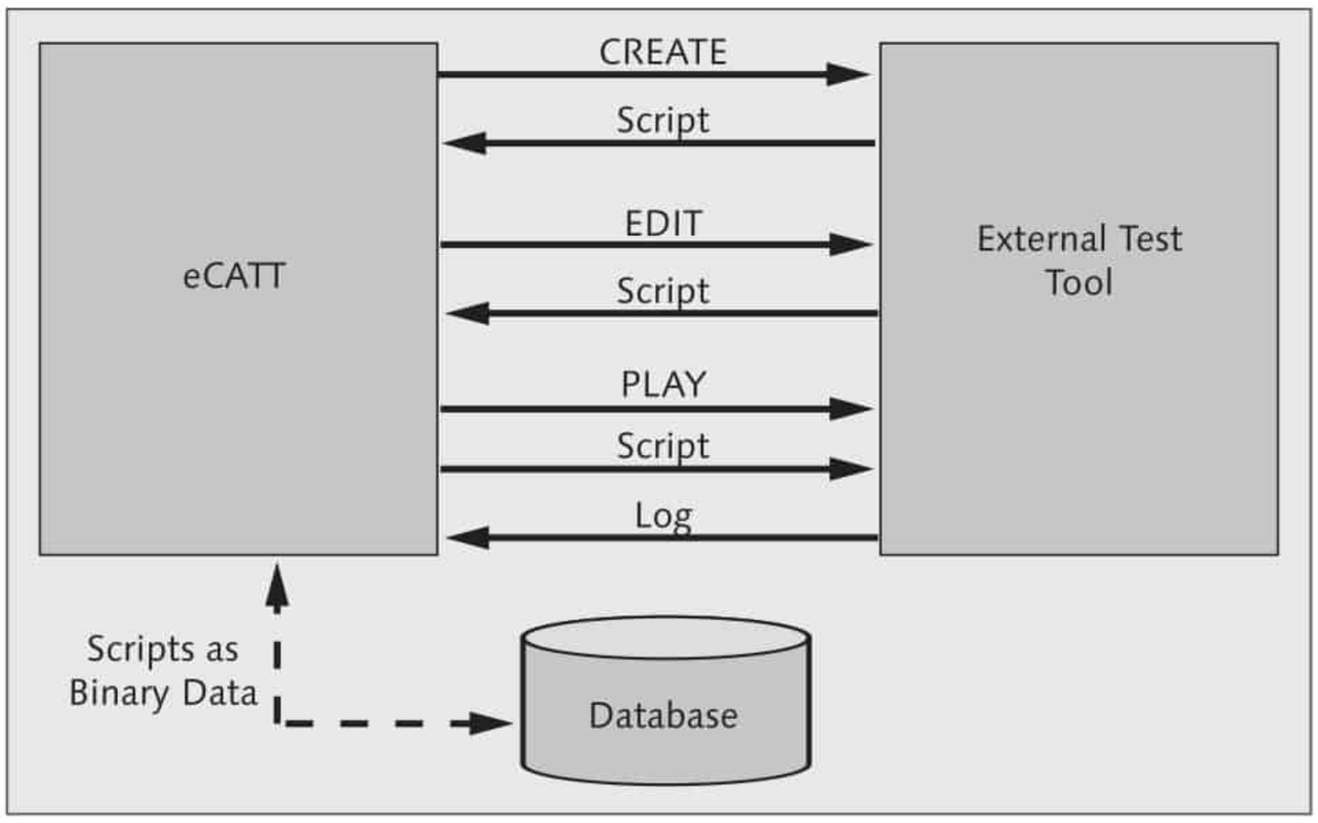


Figure 8.43 Integrated External Tool with eCATT

Integrated recording

The procedure for integrated recording is as follows.

1. When you create the script in Transaction SECATT, enter the desired test script name and select the external tool to be used.

   The system takes you to an editor window whose functions are highly reduced because the recording itself takes place in the external test tool.

2. Click the SCRIPT button to start the recording in the external tool.
3. After the interaction with the application to be tested is completed, the user stops the recording, defines transfer parameters if required, and selects the save function of the respective external test tool.
4. The recorded script is transferred via BC-eCATT and stored in the database.

Standalone recording

For standalone recording, the recording takes place in the external test tool independent of eCATT. If you want to transfer the data to eCATT after recording, then—depending on the implementation in the used external test tool—you must save the data in such a way that it can be transferred to eCATT. You then create the corresponding test script and a test configuration in eCATT.

366 Book_M.indb 466 12/7/10 3:23:40 PM

Editing

The script is edited and maintained in the external tool. For this purpose, eCATT sends the script back to the external tool. After it has been edited, the script can be uploaded again.

Execution

During the execution, eCATT interacts with the external tool in the same way, except that in this case no further user interaction is required other than starting the eCATT test configuration. eCATT sends the script to the external tool. The tool is then started by eCATT, receives the possibly transferred parameter from eCATT, and executes the script. After the script has run, result values can be returned to eCATT in the form of parameters and can then be further processed. Additionally, the log of the external tool is created, transferred to the SAP system via BC-eCATT, and stored by eCATT in the database. In contrast to the script format, the log format is defined by the BC-eCATT standard. These logs can thus be fully integrated in the log of the eCATT test configuration and be displayed, searched, and analyzed just like the logs of native eCATT scripts.

UNC path to logs

Depending on whether this function was implemented in your external test tool, you have the option in eCATT to define a Uniform Naming Convention (UNC) path to the log in the external tool. If you've defined a UNC path and a script is executed, the complete log is stored in the external tool at the location referenced by the UNC path. Additionally, a node is added to the eCATT log that specifies the UNC path as a link. If you follow this link, the log is displayed in the log viewer of the external tool. This functionality is beneficial, for example, if you store screenshots in the external system, together with the log, which are not supposed to be transferred to eCATT for performance reasons.

Calling external test drivers

The actual user interaction is not stored in the command interface when an external test driver is called. Instead, every external driver call has its own script that is marked as external. Analogous to the `REF` command that references to another test script, the external script is referenced via the `REFEXT` command.

"REFEXT" Command

The `REFEXT` command serves for calling a test script recorded by an external tool:

```
REFEXT ( <external script>,
         <command interface>,
         <version> ).
```


366 Book_M.indb 467 12/7/10 3:23:40 PM

Start options for the external driver

When an external driver is implemented, you have the following start options.

- **Normal**
  The external tool is not visible during the execution. Only the test execution, by interacting with the application to be tested, can be watched on screen.
- **Debug mode**
  The debugger of the external tool is started, and the test script stops and remains in debug mode before the first statement. Depending on the capabilities of the implemented external tool, the test script can now be executed and analyzed step-by-step in the debugger.
- **With the interface of the external tool**
  The external tool is visible during the test execution. The actual test execution is the same as for the NORMAL start option.

Start options and start profile

If the external test tool has its own authorization concept, you can store a user name and a password in the start options. These are then used to log on to the external tool. However, because you may forget to set these entries in the start options, a storage alternative is available: the start profile. If you then run a test from the Test Workbench, specify which test configuration is supposed to be executed with which start profile. A fast way to create a start profile is to copy the current settings in the start page (START OPTIONS • SAVE START PROFILE).

"SENDEXT" command

Let's assume that your script contains a `REFEXT` command with a `DO` loop, and the external script is supposed to be run 100 times. Normally, the `REFEXT` command would send the script to the external tool for every single execution. To avoid this unnecessary data transfer, the `REFEXT` command interface has the DO NOT SEND option. When you select this option, a `SENDEXT` command is put at the start so that the data is transferred only once at the beginning.

External tools and process chains

Let us now look at some implementation scenarios for the external tool driver. A typical case is the integration of a web application with an SAP system, such as the implementation of a third-party web shop as a part of a front end for an ERP system.

The key aspect from the tester's point of view is that the browser-based application of the web shop is not based on a user interface directly sup-

366 Book_M.indb 468 12/7/10 3:23:40 PM

ported by eCATT. It is out of reach for the test drivers presented so far. Because most relevant business processes also affect the web shop as an integral component, it isn't useful to exclude it from the test.

Because complete business processes are to be tested end-to-end in the context of an integration test, a separate test of the web shop is not very useful either. You need to completely and automatically test the integration with the other components of the solution landscape as well. The objective of implementing the external test tool is to close the gap in the automated test chain.

SAP-dominated solution landscape

One scenario assumes that we are dealing with an SAP-dominated system landscape with few foreign components. In this situation, an eCATT-based test using an external test tool is recommended. A major part of the testing then benefits from the deep integration of eCATT with the SAP applications; the gaps can then be elegantly closed by the external driver.

Heterogeneous system landscape with SAP share

If a system landscape is dominated by systems of one or several third-parties, SAP applications might only be used sporadically. The advantages of an eCATT-based automation solution seem less striking. Before making a decision, however, we should weigh the integration possibilities of SAP Solution Manager. In the sense of a holistic application management, it enables you to integrate steps of non-SAP applications with the Business Process Repository (BPR) and to build the test organization across systems in a process-oriented way. In addition to this organizational advantage, we should consider the chance to minimize the total cost of operation for the automation solution. For the non-integrated implementation of external test tools, in addition to the actual tool costs, there are usually costs for the infrastructure required by the tool, that is, databases, file services and their license and administration costs. With integration in eCATT, this cost can be avoided almost entirely. Even in this situation, you should always check the alternative of eCATT-based test automation.

Test tools that can be integrated

In the SAP Developer Network (SDN) on the certification page of BC-eCATT, you can obtain information on the BC-eCATT interface as well as the test tools that can be integrated. This page is available via the following link: *https://www.sdn.sap.com/irj/sdn/interface-certifications*.

366 Book_M.indb 469 12/7/10 3:23:40 PM

Conclusion

eCATT treats third-party programs implementing the BC-eCATT interface as external test tools. These external programs are addressed like native eCATT test drivers; however, they do not store the recorded information in command interfaces but in specific scripts. The external scripts themselves can be called from eCATT using the `REFEXT` command. For the standalone recording variant, the test scripts are a direct part of the test configuration. The scripts, their test data, and the execution logs are all managed within the SAP system.

8.6 Implementing Checks

An eCATT script typically consists of two logical blocks: the call of the test driver, and the subsequent checking of the results. The possibilities of recording, parameterizing, and revising commands for the various test drivers have been presented in the previous sections. We now look at the actual checking process: the comparison of the result returned by the driver and an expected value.

An important consideration is that scripts do not necessarily fail just because the recorded transaction returns an error. Conversely, a script is not necessarily successful just because the transaction returns a value. To clarify these statements, let's look at the following example:

In a transaction, a passenger is to be booked on a flight. The flight is determined by a flight number and a date. If a reservation on the specified flight is possible, it should be entered in the database, and a reservation number should be returned. The most obvious case is that a flight number and a date are entered and a reservation number is returned. However, there are a number of other possible outcomes.

Unexpected errors

1. The desired flight exists, and there is a seat available. In this case, the reservation is to be completed and a reservation number should be returned.

   If an error is still reported by the software, there might be a programming or customizing error. If this error is detected during the test, we refer to an unexpected error.

366 Book_M.indb 470 12/7/10 3:23:40 PM

Index

366 Book_M.indb 705 12/7/10 3:25:17 PM

D

E

366 Book_M.indb 706 12/7/10 3:25:17 PM

F

G

366 Book_M.indb 707 12/7/10 3:25:17 PM

H

I

J

K

L

366 Book_M.indb 708 12/7/10 3:25:17 PM

M

N

O

P

366 Book_M.indb 709 12/7/10 3:25:18 PM

Q

R

S

366 Book_M.indb 710 12/7/10 3:25:18 PM

366 Book_M.indb 711 12/7/10 3:25:18 PM

T

366 Book_M.indb 712 12/7/10 3:25:18 PM

366 Book_M.indb 713 12/7/10 3:25:18 PM

366 Book_M.indb 714 12/7/10 3:25:18 PM

U

V

W

366 Book_M.indb 715 12/7/10 3:25:19 PM

X

366 Book_M.indb 716 12/7/10 3:25:19 PM