ECATT Part 5 – Conditional Execution

Applies to:
CRM 5.0

Summary
In the Part I of eCATT series, we covered the introduction to eCATT, its prerequisites, features, when to go for SAP GUI mode recording and eCATT’s main objects. In the Part 2 of eCATT we covered just the recording and replay of the script. In the Part 3 of eCATT article learnt as to how to parameterize a script. In the Part 4 of eCATT article you saw how you can modularize a script.
In this article we will learn the conditional execution of a script, defining local variables, use of MESSAGE, END MESSAGE.

Author: Rakesh Kumar Jain
Company: Intel Technology India Pvt. Ltd.
Created on: 10 Oct 2007

Author Bio
I am Rakesh Kumar Jain, working for Intel Technology India Pvt. Ltd. I am working as a System Analyst in SAP CRM. I have worked on Interaction Center Webclient, Solution Manager. Most of my primary work is in the Service module of the CRM.
Table of Contents
Objective .......................................................................................................................................... 3
Prerequisites ................................................................................................................................ 3
Procedure ..................................................................................................................................... 3
Result: .......................................................................................................................................... 6
Related Content ............................................................................................................................... 7
Disclaimer and Liability Notice ......................................................................................................... 8
Objective

In this document, we will learn the following:

- Call another test script based on a condition
- Defining local variables
- Use of MESSAGE, END MESSAGE.

Prerequisites

You have gone through the following:
1. eCATT Part 1 - An Introduction
2. eCATT Part 2 - Recording a Scenario
3. eCATT Part 3 - Parameterization of Script
4. eCATT Part 4 - Modularizing Script

Procedure

1. Login to the SAP CRM System. Enter the Transaction = SECATT.
2. Now, create and edit the z09_DemoMasterScript. Click on Pattern Button.
   - Choose Group = Test Script References.
   - Command = REF
   - Test Script = Z09_DEMOCREATEPRODUCT.
   - Interface = Z09_DEMOCREATEPRODUCT_1.
   - Target System = None.
   Click on Continue Button.

3. Similarly add the reference to the Test Script Z09_DEMOUPDATEPRODUCT.
4. Click on Pattern.
   - Choose Group = Script Control.
   - Command = Message.
   - Interface (Type Message) = MSG_1.

5. Click on Continue Button.
6. The simplest function of the MESSAGE...ENDMESSAGE block is to collect all messages that occur from commands enclosed between the MESSAGE and ENDMESSAGE commands. Hence we place the command REF ( Z09_DEMOUPDATEPRODUCT , Z09_DEMOUPDATEPRODUCT_1 , NONE ) between the MESSAGE and ENDMESSAGE command. The Command REF ( Z09_DEMOCREATEPRODUCT , Z09_DEMOCREATEPRODUCT_1 , NONE ) would appear at the end.
7. Open the Test Script = Z09_DemoCreateProduct in the Display Mode. Click on Expand button in the Structure Editor and scroll down to the node ProcessedScreen[8] in our Demo Script. Observe the Message, ID, Number and Parameter and note them. This is the Message displayed when a product is created. We will capture this message and only if this message appears we will call our next script. Else the script will gracefully exit.

8. Go back to the Test Script = Z09_DemoMasterScript. Double click on the command interface (MSG_1) of the MESSAGE command. The command interface is displayed in the structure editor to the right of the command editor.

9. Choose and enter the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>‘R’</td>
</tr>
<tr>
<td>MSGTYP</td>
<td>‘S’</td>
</tr>
<tr>
<td>MSGID</td>
<td>‘COM_PRODUCT_WB_UI’</td>
</tr>
<tr>
<td>MSGNR</td>
<td>001</td>
</tr>
</tbody>
</table>

You have now defined a rule that requires a particular message to occur.

10. In the start options, change the error behavior to terminate when an error occurs.
11. Now, let's capture the messages that occur when the script is replayed back. Only when the Script has been run successfully and the Material is created, we will like to call our script to update the product being called. Go back to the Z09_DEMOMASTERSCRIPT and add the following lines of code.

   TOTAL_MESSAGE = &TFILL.
   CAPTURED_MESSAGE = E_MSG_1[TOTAL_MESSAGE]-MSGV1.
   LOG(CAPTURED_MESSAGE).

   The variable &TFILL would capture the total number of message captured in a script and we assign that to the variable V_MESSAGE.
   The second statement assigns the first variable part of the last message to the parameter CAPTURED_MESSAGE. For the sake of simplicity, we make the assumption that the message is always the last one. More generally, you could loop through the command interface to find the message that you are interested.
   The third statement is just a convenient command to simply show the extracted value being used.
   You will define all these variables in the Parameter List. Along with these variables you will also define a variable called ACTUAL_MESSAGE and initialize it with a value called 'Material'.

12. Declare one more variable called SKIPPED_UPDATE and Initialize it with a value = ' Did not execute Z09_DemoUpdateScript'
13. Now, we will check the Message and only on successful creation of a product, we will execute our test script, Z09_DEMOUPDATEPRODUCT. We will now use the IF..ELSE condition and compare the CAPTURED_MESSAGE with the ACTUAL_MESSAGE. If they dont match then we just print a value called ' Did not execute Z09_DemoUpdateScript'.
   The complete code now should look something like this:

   MESSAGE ( MSG_1 ).
   REF ( Z09_DEMOCREATEPRODUCT , Z09_DEMOCREATEPRODUCT_1 , NONE ).
   ENDMESSAGE ( E_MSG_1 ).

   TOTAL_MESSAGE = &TFILL.
   CAPTURED_MESSAGE = E_MSG_1[TOTAL_MESSAGE]-MSGV1.
   LOG (CAPTURED_MESSAGE).

   IF CAPTURED_MESSAGE = ACTUAL_MESSAGE.
      REF ( Z09_DEMOCREATEPRODUCT , Z09_DEMOCREATEPRODUCT_1 , NONE ).
   ELSE.
      LOG ( SKIPPED_UPDATE ).
   ENDIF.

14. You are now ready to execute the above script conditionally.
Result:

1. When a product is created newly, the log message would look something like this on successful execution.

```
+ Z09_DEMOMASTERSCRIPT [02:08 min] Version 1 Demo Master Script
  ▶ IMPORT Z09_DEMOMASTERSCRIPT 05:01:09
  ▶ LOCAL VARIABLES
  ▶ MESSAGE MSG_1 [01:32 min]
  ▶ ENDMESSAGE E_MSG_1 (&TFILL = 1) = XML-DATA-01
  ▶ = TOTAL_MESSAGE = &TFILL
  ▶ = CAPTURED_MESSAGE = E_MSG_1[TOTAL_MESSAGE]-MSGV1
  ▶ LOG CAPTURED_MESSAGE = Material
  ▶ IF CAPTURED_MESSAGE = ACTUAL_MESSAGE
    ▶ ENDIF
  ▶ EXPORT Z09_DEMOMASTERSCRIPT 05:03:13
```

2. When a product creation fails for some reason, the next script Z09_DEMOUPDATESCRIPT is not executed and the log message would look something like this on execution.

```
+ Error in eCATT command ENDMESSAGE
  Requested Message S COM_PRODUCT_WB_UI 001 Did Not Occur
  ▶ IMPORT Z09_DEMOMASTERSCRIPT 05:21:22
  ▶ LOCAL VARIABLES
  ▶ MESSAGE MSG_1 [01:00 min]
  ▶ ENDMESSAGE E_MSG_1 (&TFILL = 1) = XML-DATA-01
    ▶ Requested Message S COM_PRODUCT_WB_UI 001 Did Not Occur
  ▶ = TOTAL_MESSAGE = &TFILL
  ▶ = CAPTURED_MESSAGE = E_MSG_1[TOTAL_MESSAGE]-MSGV1
  ▶ LOG CAPTURED_MESSAGE =
  ▶ IF CAPTURED_MESSAGE = ACTUAL_MESSAGE
  ▶ ELSE
    ▶ LOG SKIPPED_UPDATE = Did not execute Z09_DemoUpdateScript
  ▶ ENDIF
  ▶ EXPORT Z09_DEMOMASTERSCRIPT 05:22:24
```
Related Content

http://help.sap.com/saphelp_nw2004s/helpdata/en/8e/df9f40eb72371be10000000a1550b0/frameset.htm
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.