Ready to Start Leveraging eCATT? Take Your Existing CATT Tests with You!

Jonathan Maidstone

Jonathan Maidstone joined SAP in 1996 as a translator for the ABAP Language and ABAP Workbench departments. After a spell as a technical writer, during which he was involved with the SAP Control Framework and DCOM Connector projects, he is now a product management specialist with particular responsibility for rolling out eCATT, SAP's new test tool.

(complete bio appears on page 24)

Many of you are familiar with SAP's Computer Aided Test Tool (CATT) and have used it to test your SAP implementations, upgrades, and support packages since its introduction in R/3 Release 3.0. Since then, SAP and its customers and partners have created an enormous number of CATT test procedures, many of which are still in active use in automated test projects. At the same time, technological and strategic developments in the SAP world have pushed the "typical" SAP installation (if such a thing exists) farther away from the centralized, traditional dynpro environment for which CATT was originally designed, making testing with CATT increasingly difficult. In order to close the gaps opened up by these changes, SAP has introduced the *extended* Computer Aided Test Tool (eCATT) as part of SAP Web Application Server (Web AS) 6.20.[1] eCATT now provides testing teams with:

- **Support for the EnjoySAP user interface controls:** With the EnjoySAP initiative, the user interfaces of many applications were redesigned with PC-based controls, such as the SAP Tree Control, ALV Grid Control, and SAP HTML Viewer, to make the applications easier to navigate and operate. eCATT enables you to test such controls-based applications, which could not be tested with CATT.[2]

[1] eCATT is a new tool accessed with transaction code *SECATT*. For complete details on what's changed between CATT and eCATT, see my article "Extend the Range and Reduce the Costs of Your SAP Testing Activities with eCATT" in the January/February 2003 issue of this publication.

[2] CATT procedures are analogous to BDC (business data communication) scripts in that they simulate user actions by passing data and transaction function code in sequence to the transaction being tested. Since the EnjoySAP controls in 4.6C handle many processing tasks, like paging and sorting, locally on the frontend PC (i.e., they do *not* trigger transaction function code), CATT has no way of simulating these actions and thus cannot adequately test many 4.6+ programs.

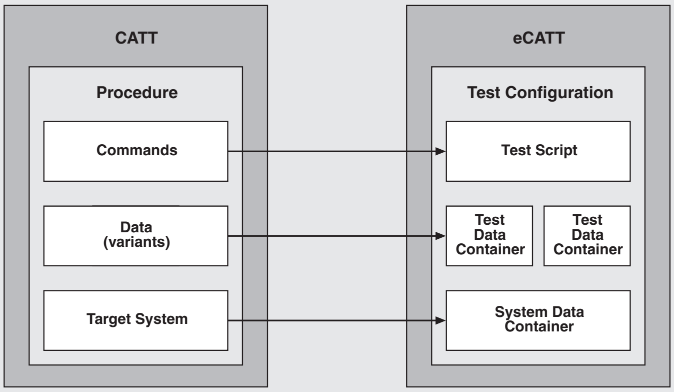
Understanding eCATT's New Object Model

At a high level, you need to have three elements in place to conduct a test in your SAP system:

- **Commands** that specify which applications to run, how to navigate within them, and the required follow-up actions
- **Data** to be entered in application screens or used internally to control the test, including variants (sets of different test parameters)
- **System connection information** for accessing the systems you want to remotely test

In CATT, all three of these elements were defined in a single object called a *procedure*, as shown in the diagram below. In eCATT, however, they have been separated into their own objects, which allows data and system connection information to be reused across multiple scripts and remain centrally maintained.

As far as terminology, the term "procedure" has been retired. In eCATT, commands are now stored in *test scripts*, reusable test data is stored in *test data containers*, and system connection information is stored in *system data containers*. A new object has also been introduced, a *test configuration*, which contains all of the elements required to implement a test: a test script, test data containers, and a system data container.

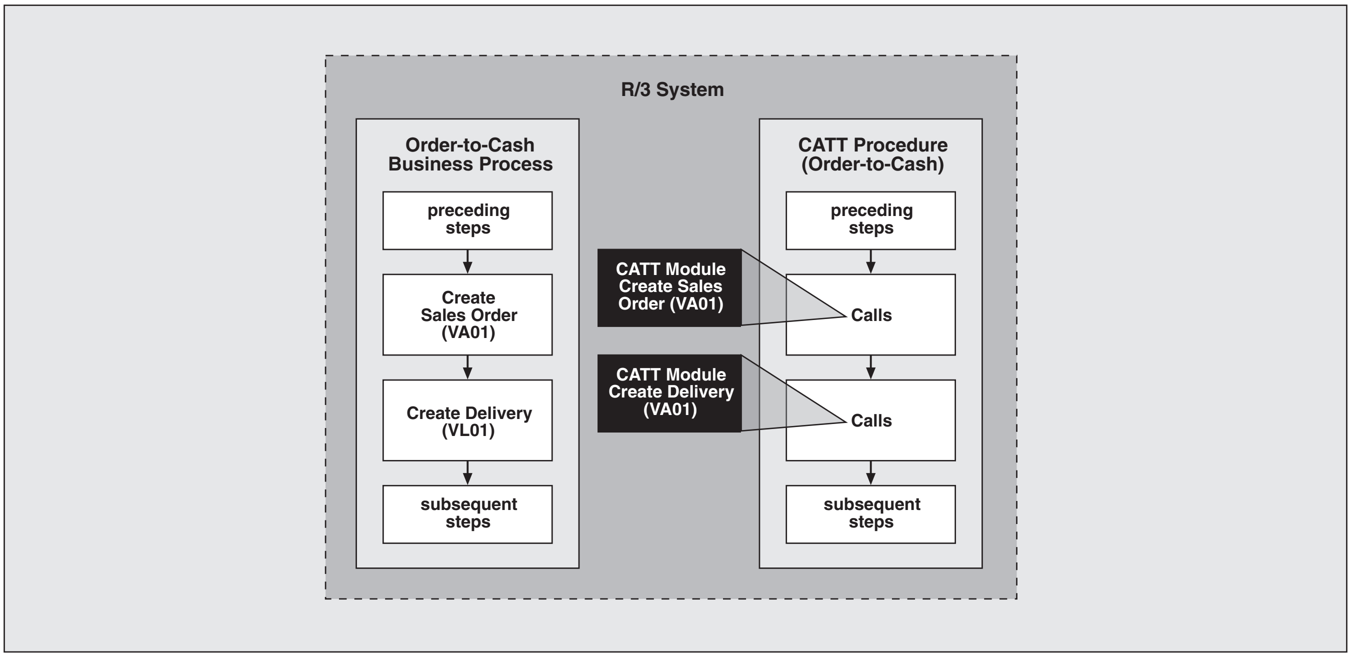


- **Support for distributed, remote testing:** While the eCATT "master" test script must run on a 6.20+ system, you can remotely call test procedures and scripts located on any other SAP systems in your environment, as long as they are running 4.6C or higher. This allows you to easily test business processes that span multiple heterogeneous systems, such as R/3 and CRM (Customer Relationship Management). It is also possible to configure CATT to run tests across distributed systems; however, this is difficult and the capabilities are limited because CATT was designed for centralized R/3 installations.
- **The ability to test web applications:** eCATT enables integration with certified third-party testing tools for testing web applications built with Web AS and ITS (Internet Transaction Server) technology or created using third-party tools,[3] which was not possible with CATT.

In addition to these enhanced capabilities, eCATT is loaded with brand-new features that make creating test scripts even easier and more efficient, including:

[3] SAP's BC-eCATT interface certification verifies that a third-party product can integrate with eCATT. Currently, Compuware's TestPartner is the only such certified product. For more information about interface certification, visit **www.sap.com/icc**.

Figure 1 *Company X's R/3-Based Order-to-Cash Process and the Corresponding "Master" CATT Procedure*


R/3 System
Order-to-Cash Business Process
preceding steps
Create Sales Order (VA01)
Create Delivery (VL01)
subsequent steps
CATT Module Create Sales Order (VA01)
CATT Module Create Delivery (VA01)
CATT Procedure (Order-to-Cash)
preceding steps
Calls
Calls
subsequent steps

- **A new, more flexible object model:** With eCATT's new model, test data can be stored and used across test scripts, and remote systems are referenced via symbolic "logical" names instead of technical RFC (Remote Function Call) names (see the sidebar on the previous page for more on the new eCATT model and terminology).

- **A new, easier-to-read command editor:** eCATT test scripts now read more like traditional programming code, instead of the sequential list of command/attribute pairs displayed in CATT.

- **A new user interface:** eCATT's state-of-the-art user interface is very easy to learn and use.

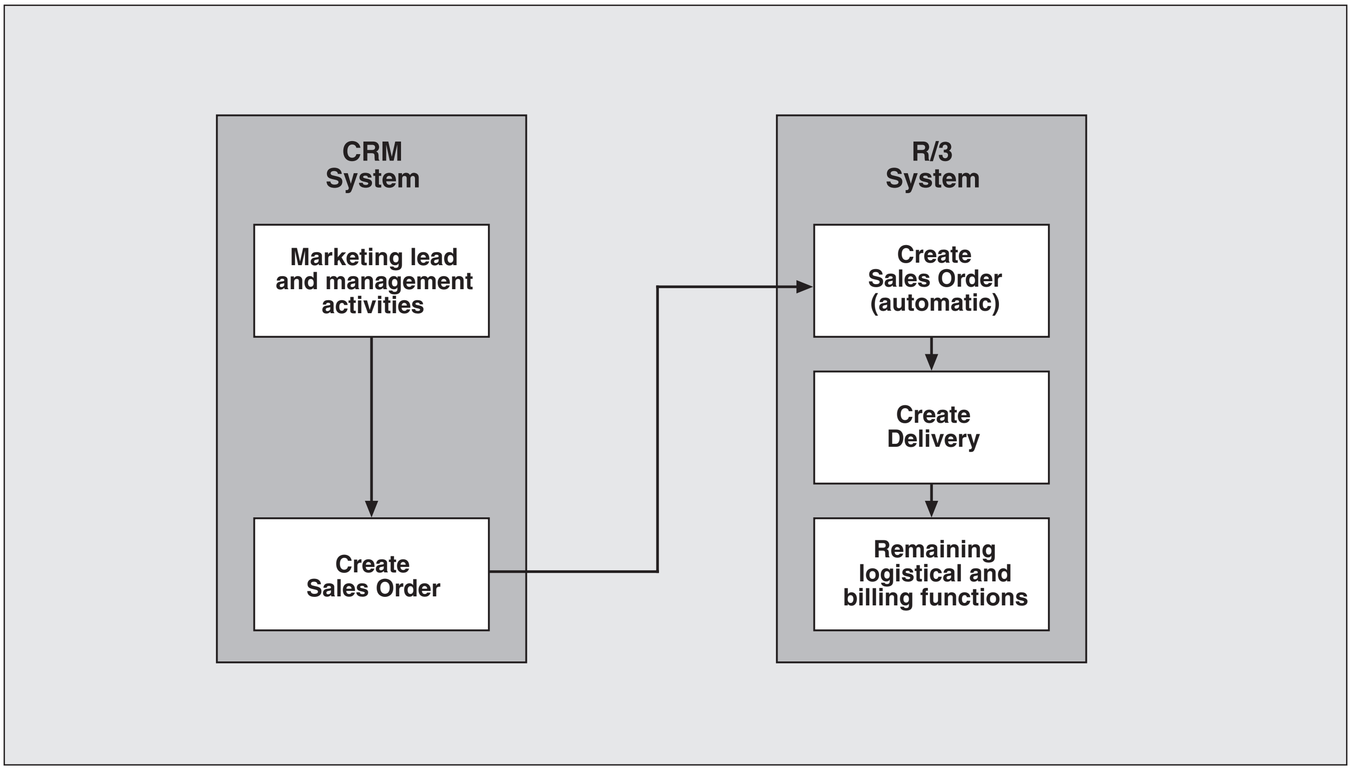
So, while all of this new functionality sounds great, what about your current CATT test objects? If your company is like most, you have already made a significant investment in writing test procedures, setting up test data, creating test modules, and so on. Is this investment lost? Absolutely not! This article shows testing teams how to continue to use existing CATT objects by migrating them all to eCATT (using a built-in migration tool), leaving them as-is in 4.6C and calling them remotely from eCATT, or doing both (the option most are likely to choose). By the time you've finished reading this article, you'll know how to simultaneously safeguard your current CATT investment and put yourself in a position to leverage eCATT's new capabilities.

Which CATT Objects Should I Migrate?

Throughout most of this article, for illustration purposes I will use a single example scenario, which I hope will be true-to-life for some, and easy to follow for everyone else. Consider the case of Company X, which has decided to implement mySAP CRM 3.0 in addition to their existing 4.0B SAP R/3 installation. In the past, their order-to-cash business process ran entirely in R/3, and because they were running 4.0B, they used CATT to automate their testing (remember that CATT was introduced as part of R/3 3.0).

Take a look at **Figure 1**. It depicts a small piece

Figure 2 *The Redesigned Order-to-Cash Process Split Between the CRM and R/3 Systems*


CRM System
Marketing lead and management activities
Create Sales Order
R/3 System
Create Sales Order (automatic)
Create Delivery
Remaining logistical and billing functions

of Company X's existing order-to-cash business process. The left side shows the process flow, and the right side shows the CATT objects being used to test it: a single "master" test procedure and two test modules that call transaction *VA01* (Create Sales Order) and *VL01* (Create Delivery), respectively.

Now comes the change. Suppose Company X decides to upgrade from R/3 4.0B to 4.6C,[4] and at the same time wants to begin creating sales orders in the newly installed 3.0 CRM system instead of in the R/3 system. Because of the tight integration between CRM and R/3, all orders created in the CRM system are automatically mirrored in the R/3 system. The result is shown in **Figure 2**.

So, as members of Company X's testing team, how are we going to implement testing in this new multisystem process? Before we only had to deal with a single R/3 system. Now we have a process that spans two systems!

One idea is to simply modify our existing CATT test module for the Create Sales Order transaction (*VA01*) so that it is CRM-compatible, and then specify an RFC destination for it that points to the CRM system. Unfortunately, this won't work, because the CRM-based Create Sales Order transaction relies heavily on the EnjoySAP controls, which CATT cannot test.

Another idea is to write an eCATT test script on the CRM box and modify the CATT procedure to call this script instead of the local *VA01* module. But this won't work either — while you can call CATT modules and procedures from eCATT, you can't call eCATT scripts from CATT. And furthermore, eCATT can only run on 6.20 or higher systems (remember, our example CRM system is running 6.10).

[4] The release upgrade is critical to the example, because eCATT scripts are able to remotely call "as-is" CATT procedures only if they are located in a 4.6C or higher system.

Should I Just Migrate All of My CATT Objects to eCATT Right Away?

Sooner or later, most organizations will likely upgrade their R/3 systems to 4.6C or higher, or implement a complementary system like CRM (Customer Relationship Management) or APO (Advanced Planner and Optimizer). So the more meaningful question is *when* to migrate your existing CATT objects, rather than *whether* to migrate them: Should you do it all at once? One "test" at a time? One module at a time?

While the cleanest approach is to consolidate all of your objects at once into a single tool (i.e., eCATT) via a "big-bang" approach, it's probably not a viable option for you due to time and resource constraints. Don't worry, though! CATT and eCATT run side by side in Release 6.20, so there is no pressing need to migrate everything all at once.

Instead, you can write your new test scripts in eCATT and continue using your existing CATT procedures for as long as they still work, replacing them with eCATT scripts as necessary. If you have central CATT objects called by many procedures, you may decide to migrate the called objects in smaller groups, one subject area at a time. Remember, though, that a CATT procedure cannot call an eCATT script, so if you migrate a CATT object to eCATT, you must also migrate any CATT procedures that call it. Conversely, if you have a test object that is called by both a procedure to be migrated to eCATT and one that you do not plan to migrate yet, hold off on migrating the test object until you are ready to migrate both procedures.

✓ *Tip*

To avoid ending up with a mix of CATT and eCATT test objects, you can simply migrate all of your CATT objects to eCATT if you want. See the above sidebar for help on making this decision.

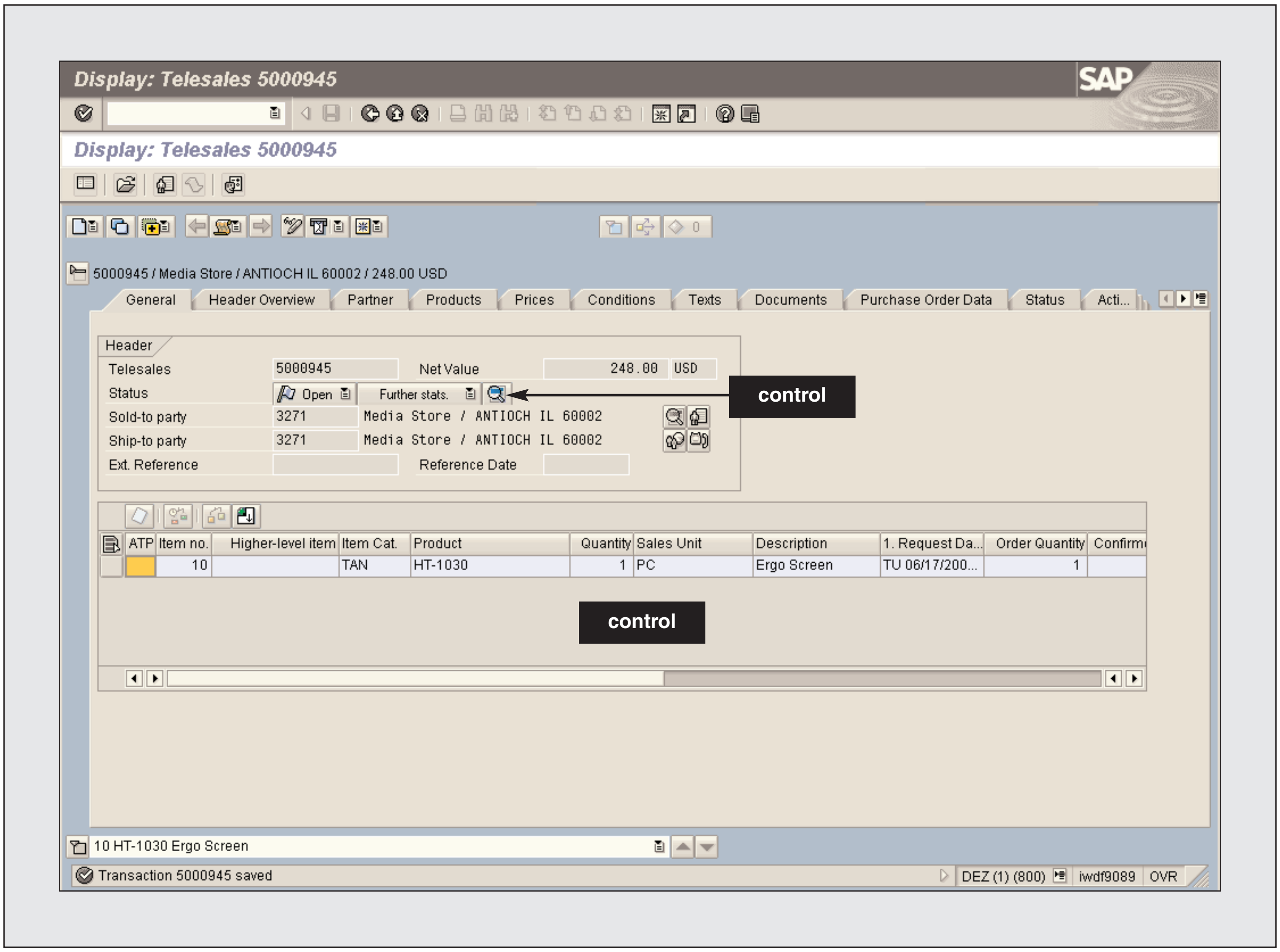
This leads us to our third option: run the test from a "master" eCATT script on a Web AS 6.20 box, which we know will work.[5] But will we then have to redevelop *all* of our CATT procedures and modules from scratch in eCATT? Thankfully, no! We can migrate some to eCATT and leave others as-is in the R/3 4.6C system and call them remotely. Let's figure out what we want to migrate and what we want to leave as-is.

[5] Since CRM 3.0 (the version implemented by our example company) runs on Web AS 6.10, eCATT, which runs on 6.20, is not a part of the CRM system. In the next section on preparing the system landscape, you will see how you can locate the central eCATT script so that this is not a problem.

Let's first consider the CATT test module for the Create Delivery transaction (*VL01*). Since we will still run this transaction in R/3 (and inspection of the module reveals that the test actions involved do not interact with any unsupported user interface controls[6]), we can leave the *VL01* module as-is in the R/3 4.6C system and call it remotely from eCATT. In contrast, the test module for the Create Sales Order

[6] It is possible to use CATT to test transactions that use EnjoySAP controls, such as the tree control or the dropdown list box control, as long as the test actions do not require interaction with any of these controls (e.g., the module launches the transaction, enters data in a few regular text fields, navigates using standard pushbuttons and menus, and simply saves).

Figure 3 *A Create Sales Order Screen in CRM That Uses New User Interface Controls Unsupported by CATT*



Display: Telesales 5000945

5000945 / Media Store / ANTIOCH IL 60002 / 248.00 USD

General | Header Overview | Partner | Products | Prices | Conditions | Texts | Documents | Purchase Order Data | Status | Acti...

Header

Telesales	5000945	Net Value	248.00 USD
Status	Open	Further stats.	
Sold-to party	3271	Media Store / ANTIOCH IL 60002	
Ship-to party	3271	Media Store / ANTIOCH IL 60002	
Ext. Reference		Reference Date	

ATP	Item no.	Higher-level item	Item Cat.	Product	Quantity	Sales Unit	Description	1. Request Da...	Order Quantity	Confirm
	10		TAN	HT-1030	1	PC	Ergo Screen	TU 06/17/200...	1	

10 HT-1030 Ergo Screen

Transaction 5000945 saved

DEZ (1) (800) iwdf9089 OVR

transaction (*VA01*), which now runs in the CRM system, will call a CRM-based screen that employs several EnjoySAP controls that require interaction (see **Figure 3**). As CATT does not support the testing of controls, we must either migrate the *VA01* test module from CATT to eCATT or re-create it from scratch in eCATT.

But what about the original "master" CATT procedure? As mentioned previously, you can't call eCATT scripts from CATT, so we won't be able to call our new eCATT objects remotely if the master is running in CATT. Therefore, we need to migrate the master procedure as well. Don't worry, though; you'll soon see how easy this is using eCATT's built-in migration tool.

✓ ***Note!***

As you might imagine, the CRM-based Create Sales Order screens are much different than the 4.0B screens our CATT module was originally built to test. So should we go through the effort of migrating the test module, or should we simply re-create it from scratch in eCATT? Usually, the answer is to migrate the module, since migrating preserves its attributes, interface definition (importing and exporting parameters and local variables), and any coding for calculated fields, and saves us from having to redefine these elements and risk introducing errors.

Figure 4 Migration Plan for Company X's Order-to-Cash Test Objects

CATT Test Object	Action Required
Create Sales Order (VA01)	Migrate to eCATT.
Create Delivery (VL01)	None, although migration to eCATT is possible.
Master procedure for testing the process	Migrate to eCATT.

Figure 4 summarizes what we will have to do to update our test process to be run in eCATT, but before we get started, we need to make sure we have the right system landscape in place.

Prepare the System Landscape

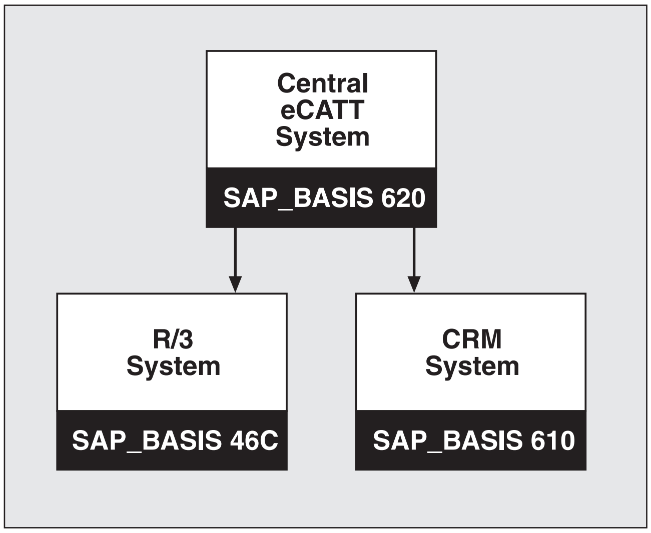
Before performing a migration, you need to consider the system landscape in which the testing will be performed.

Let's look at our example company. Previously, Company X had a single R/3 installation and ran tests locally within that system. Now there are two components — an R/3 system and a CRM system. Given that the R/3 system is running 4.6C and the 3.0 CRM system is running 6.10, neither is suitable for hosting our new eCATT test scripts, which require 6.20.

Figure 5 illustrates a solution for this situation — you can install a standalone 6.20 instance to serve as a central test system in which you can create and maintain all of your eCATT objects.[7] From this central system, you can run tests on any system in your landscape running 4.6C or higher, using the new eCATT commands *REFCATT* and *REMOTECATT* (these commands are discussed in detail in the next section, "Perform the Migration").

With the system landscape prepared, we are ready to migrate our example CATT test module for the

Figure 5 A System Landscape for eCATT Testing



Create Sales Order transaction (*VA01*) along with our master CATT procedure.

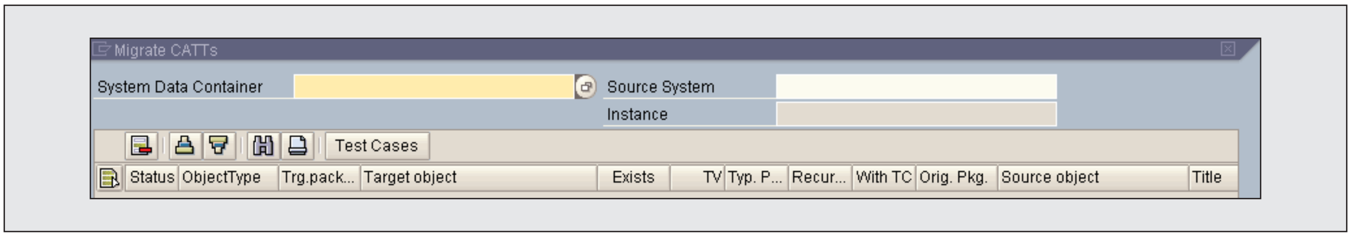
Perform the Migration

As I mentioned earlier, migrating CATT procedures and test modules to eCATT is easy using eCATT's built-in migration tool.

Two major things happen during the migration. First, a new eCATT test script is created in the target system (in our example scenario, the central 6.20 system). Second, the old CATT commands are translated into new eCATT commands (see the "Download Files" section at **www.SAPpro.com** for a full listing of these translated commands). This translation is similar to translating between British and American

[7] Consider this solution only if you do not yet have a 6.20 Web AS underpinning one of your existing SAP solutions (such as R/3 Enterprise), or if you decide that you want to run your testing operations from a dedicated central instance.

Figure 6 **The eCATT Migration Dialog**



Migrate CATTs

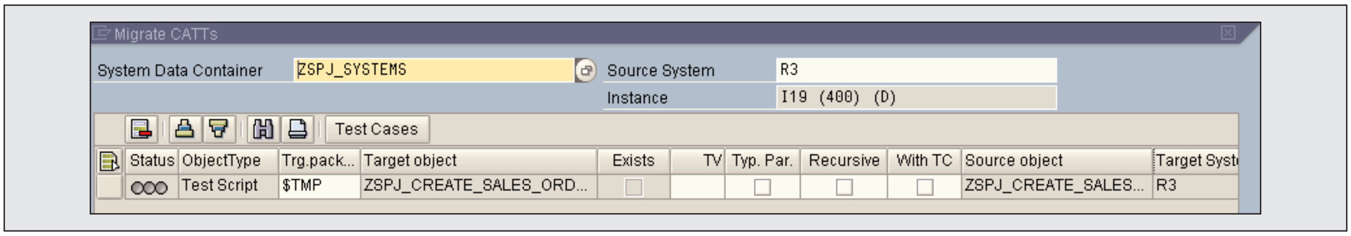
System Data Container:

Source System:

Instance:

Status	ObjectType	Trg.pack...	Target object	Exists	TV	Typ. P...	Recur...	With TC	Orig. Pkg.	Source object	Title

Figure 7 **Ready to Migrate**



Migrate CATTs

System Data Container: ZSPJ_SYSTEMS

Source System: R3

Instance: I19 (400) (D)

Status	ObjectType	Trg.pack...	Target object	Exists	TV	Typ. Par.	Recursive	With TC	Source object	Target Syst
○○○	Test Script	$TMP	ZSPJ_CREATE_SALES_ORD...	☐		☐	☐	☐	ZSPJ_CREATE_SALES...	R3

✓ *Tip*

The eCATT migration tool leaves the original CATT procedure or test module untouched (it simply creates a new eCATT test script with the same name[8]*), so if anything goes wrong, simply retry the migration. You can also continue to use the original test object in your old CATT master procedures if you want. If you no longer need the original test object, after migrating it and testing it a few times (to make sure everything works properly), you can delete it manually from CATT.*

✓ *Important!*

Before beginning the migration, make sure you've performed all of the actions described in the sidebar "eCATT Migration Prerequisites" on the next page.

English — most words are the same, but there are a few that have subtle differences.

To demonstrate the migration process, let's migrate our CATT module for the Create Sales Order transaction (*VA01*).

Migrate the CATT Test Module

We begin the migration of the Create Sales Order module from within eCATT (transaction *SECATT*) in our central test system (refer back to Figure 5) by choosing *eCATT Object* → *Migrate CATT*. A dialog box appears (see **Figure 6**), which we will use to control the process and specify migration options.

Next, we specify the CATT object we want to migrate. In our example, the Create Sales Order module is located in a remote R/3 system, so we start by specifying the eCATT system data container we created for this particular migration when we prepared the system landscape (*ZSPJ_SYSTEMS* in the example), which specifies the symbolic "logical" name of the system containing the module we want to migrate (*R3* in our example scenario) and its connection information. We then enter this logical system name in the *Source System* field (see **Figure 7**).[9] The value of the *Instance* field is prepopulated by the system.

[8] Although they have the same names and are technically stored in the same repository, the objects don't conflict, because they have different object types (i.e., "CATT procedure" vs. "eCATT test script").

[9] If you are migrating local CATT procedures, leave the *System Data Container* and *Source System* fields blank.

eCATT Migration Prerequisites

Before launching the eCATT migration tool, make sure you have done the following:

- ☑ **Set up your system data containers:** During the migration, you'll need to identify the system in which your source CATT objects are located. The eCATT migration dialog relies on the system data containers you've set up to build the list of systems to choose from when you specify the migration options. For a refresher on how to set up system data containers, see the sidebar "Working with eCATT System Data Containers" below.
- ☑ **Check your object's syntax:** Before you migrate a CATT object, ensure that it is syntactically correct using CATT's integrated syntax check. Errors can be difficult to find afterward! The potential problem here is that the CATT runtime is sometimes more forgiving than its eCATT counterpart. For example, if you erroneously specify a four-digit message number in a command (only three digits are allowed), CATT will interpret the first three and simply ignore the fourth. eCATT, on the other hand, will return an error and tell you that four-digit message numbers are not allowed.
- ☑ **Open and close your CATT object:** If you are migrating a CATT object that you have not modified since upgrading to Release 4.6C, you must open it in CATT, save it, and close it before migrating it. This is due to an internal technical conversion that was implemented in CATT in 4.6C. The conversion is performed automatically whenever you open a CATT object for the first time in a 4.6C system, and it is a prerequisite for successful migration to eCATT.

Working with eCATT System Data Containers

Unlike in CATT, in eCATT you do not address target systems directly using RFC (Remote Function Call) destinations. Instead, you create a map of the system landscape called a *system data container*. Each entry in a system data container relates to a single system in the test landscape (e.g., you would make one entry for your R/3 system, another for your CRM system, and so on). Each system entry consists of a symbolic "logical" name, by which the system is addressed from the eCATT scripts, and a corresponding RFC destination, which describes in technical terms where to find the system and how to log on to it (e.g., client, language, and user information).

This additional layer of abstraction makes eCATT scripts more flexible, because you can execute them against different systems, or even different system landscapes, by merely linking the script to a different system data container in which the logical names remain the same, but the RFC destinations are different.

To migrate from CATT to eCATT, use a single entry in a system data container to identify the system from which the CATT object will be taken.

For more details on setting up system data containers in eCATT, visit the SAP online help at **http://help.sap.com** and select *SAP NetWeaver → SAP Web Application Server 6.20 → SAP Library → SAP NetWeaver Technology Components → SAP Web Application Server → Computer Aided Test Tool → eCATT: extended Computer Aided Test Tool → Using eCATT → System Data Editor → Creating System Data Containers*.

✓ ***Tip***

If you forget the name of the system in which the CATT object resides, place the cursor in the Source System field, press F4, and choose from the list of systems defined in the system data container.

Continue by clicking the *Test Cases* button on the toolbar. A "possible values" dialog (not shown) presents a list of CATT objects available in the source system. In the example here, we choose our module for the Create Sales Order transaction, *ZSPJ_CREATE_SALES_ORDER* (see Figure 7).

We now specify the migration options. The following six fields (shaded gray in Figure 7) are prepopulated by the system and cannot be changed:

- **Status:** At this initial stage, the indicators of the traffic light status icon are gray. After the migration, the icon will display a green indicator to signify success, a yellow indicator to signify a warning, or a red indicator to signify an error.
- **ObjectType:** This is always a test script.
- **Target object:** The name that the object's new eCATT script will have. (This is always identical to the name of the original CATT object.)
- **Exists:** If this flag is checked, there is already an existing eCATT script with the target name.[10] In this case, the migration tool will let you proceed only if you specify a unique version number[11] in the target version field (more on this field in a moment, when I discuss the editable fields).
- **Source object:** The name of the original CATT object that you are migrating.
- **Target System:** The system in which the new eCATT script will be executed. (By default, the system prepopulates this field with the same value as the source system, so we will need to modify it to enable the script to run the transaction locally in the new system; more on this shortly.)

The following five fields (shown in white in Figure 7) can be modified to control the migration:

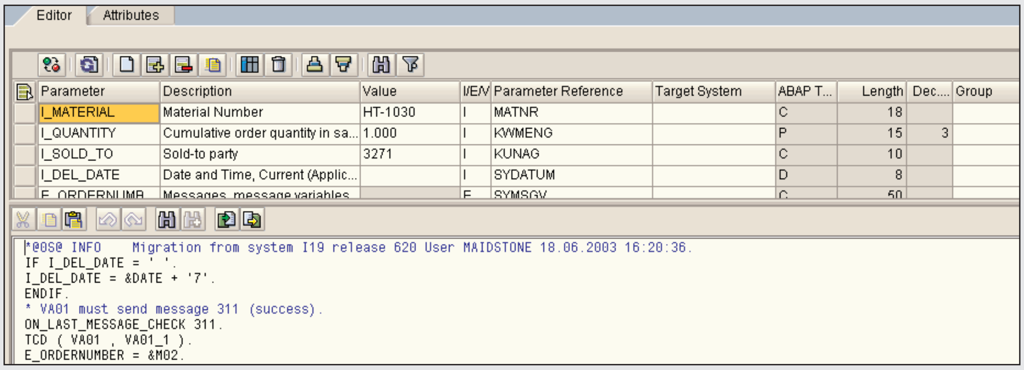
- **Trg.pack... (target package):** As eCATT objects are Repository objects, they must be assigned to a package.[12] For convenience, we use a generic *$TMP* package in the example. In practice, you will set up a separate package for each application area or team to ease administration (in case you need to transport your test objects) and access (eCATT authorizations can be restricted by package, among other factors).
- **TV (target version):** If the *Exists* field described previously is checked, you must enter an unused version number here. If you don't specify a number, eCATT assumes you want to create version 1, or prompts you for another if that already exists.
- **Typ. Par. (typed parameters):** In CATT, parameters were all preceded by the ampersand symbol (&). In eCATT, this symbol is reserved for special eCATT variables, such as those for the current date or system ID (see the "Download Files" section at **www.SAPpro.com** for a list of select CATT variables and their eCATT equivalents). By default, eCATT parameters have the same names as the original CATT parameters, but they are preceded by an underscore (_) rather than an ampersand. Thus the CATT parameter *&ORDER_NUMBER* would become *_ORDER_NUMBER* in eCATT. If you check the typed parameters box, however, the new eCATT parameters are created with prefixes that identify their function (import, export, or variable). Therefore, import parameter *&MATERIAL* would be rendered as *I_MATERIAL*, export parameter *&ORDERNUMBER* as *E_ORDERNUMBER*, and local variable *&DAY* as *V_DAY*.

[10] This might happen if you migrate a CATT object to eCATT, make further changes to the original object in CATT, and then want to re-migrate the updated CATT object, without losing the first migrated version. Versioning in eCATT allows you to follow a script's change history.

[11] For details on eCATT versioning, go to **http://help.sap.com** and select *SAP NetWeaver → SAP Web Application Server 6.20 → SAP Library → SAP NetWeaver Technology Components → SAP Web Application Server → Computer Aided Test Tool → eCATT: extended Computer Aided Test Tool → General Information → Test Script Versions*.

[12] The term *package* is used in the same context here as in the ABAP world — a package groups objects together for administrative purposes. Packages used to be called *development classes*.

Figure 8 *The Migrated Create Sales Order Test Module*



Parameter	Description	Value	I/E/V	Parameter Reference	Target System	ABAP T...	Length	Dec....	Group
I_MATERIAL	Material Number	HT-1030	I	MATNR		C	18		
I_QUANTITY	Cumulative order quantity in sa...	1.000	I	KWMENG		P	15	3	
I_SOLD_TO	Sold-to party	3271	I	KUNAG		C	10		
I_DEL_DATE	Date and Time, Current (Applic...		I	SYDATUM		D	8		
E_ORDERNUMB	Messages, message variables		E	SYMSGV		C	50		

```
*@0S@ INFO    Migration from system I19 release 620 User MAIDSTONE 18.06.2003 16:20:36.
IF I_DEL_DATE = ' '.
I_DEL_DATE = &DATE + '7'.
ENDIF.
* VA01 must send message 311 (success).
ON_LAST_MESSAGE_CHECK 311.
TCD ( VA01 , VA01_1 ).
E_ORDERNUMBER = &M02.
```

✓ *Note!*

Whether you choose to use the typed parameters feature or not is a matter of personal preference, but either way you should make the decision as a team and stick to one approach. Otherwise, you run the risk of your scripts being unable to communicate with each other because one is looking for an _ORDER_NUMBER parameter while the other uses I_ORDER_NUMBER!

- **Recursive:** This option allows you to specify how test objects referenced by the master test script should be treated during the migration. If we had selected the master CATT procedure instead of the *VA01* module for migration, and then checked *Recursive* here, the migration tool would automatically migrate the modules referenced by the master procedure in addition to the master procedure itself. We will look at this in more detail in the later section "Recursive Migration: A Shortcut for Migrating Multiple CATT Objects to eCATT."

- **With TC (test configuration):** If you check this box, an eCATT test configuration containing all of the test data, any test data variants, and the system connection information is created along with the eCATT test script. This is useful if you later want to assign variants to the script to run different versions of the test.

✓ *Note!*

If the original CATT procedure contains variants, a test configuration will be created automatically to accommodate them whether you check the "With TC" option or not.

Once we have confirmed all of these settings, we start the actual migration by clicking on the continue button (✓) at the bottom of the dialog box. After some processing, eCATT changes the traffic light icon in the *Status* field to signify success or failure. If the process succeeds, your new eCATT script awaits. If the process fails, check that the CATT object actually exists, make sure you were looking for it in the correct system, and then try the process again (remember, the original CATT object remains untouched by the migration process).

The migrated version of our Create Sales Order test module is shown in **Figure 8**. The grid at the top lists the script parameters, and the script code appears in the editor below, which you can use to make any future changes to the script (see the sidebar "Migrating Date Fields" on the next page for one type of modification you may need to make right away).

If you compared the new code shown in Figure 8 to that of the original CATT procedure, you would

Migrating Date Fields

An area in which you may need to exercise a little caution during migration is the handling of date fields. In CATT, you could use either date or character values as parameters for a date field. With date values, the date was always stored in the SAP internal format *yyyymmdd*. With character values, the date was converted into the country-specific format stored in your user settings (for example, *mmddyyyy* for the US).

In eCATT, dates are *always* stored in the internal format, regardless of whether a character or date value is used in the field. Using character values can cause problems in the following scenarios:

- **When a date value is passed to a screen field:** In this case, the value is placed in the screen field in the internal format with no conversion, which will lead to errors as the system tries to interpret the contents of the field according to your local PC's format. So, for example, the date June 20th 2003 will be rendered as *20030620* by eCATT in the SAP system. In Germany, a user's settings would interpret this as *20.03.0620* — technically a valid date, but certainly not the one you intended! In the US, the interpretation would be *20/03/0620*, which will cause an error because, of course, there is no 20th month in the calendar year.

Figure 9 Commands Changed by the Migration Tool in the Example Create Sales Order Module

CATT	eCATT
CHEERR (311)	ON_LAST_MESSAGE_CHECK 311.
TCD VA01	TCD (VA01, VA01_1).
SETVAR &ORDERNUMBER = &M02	E_ORDERNUMBER = &M02.

notice the following differences, which are summarized in **Figure 9** (again, more details on how commands have changed between CATT and eCATT are available for download at **www.SAPpro.com**):

- The CATT command *CHEERR*, which checks for a system message, has been replaced with the new command *ON_LAST_MESSAGE_CHECK* in eCATT, which has the same function.[13]
- In CATT, the field list of the TCD command, which records the values to be input in a transaction's fields during a test, has no explicit name. In eCATT, this same field list is converted into a specific command interface,[14] which we can see as the second argument in the TCD command in Figure 8 (*VA01_1* in the example).

[13] In Release 6.40, *ON_LAST_MESSAGE_CHECK* will be superseded by a new message concept, the details of which you can find in the article "Extend the Range and Reduce the Costs of Your SAP Testing Activities with eCATT" (*SAP Professional Journal*, January/February 2003, page 106).

[14] In eCATT, any command that requires data to be passed to and from it has a *command interface*, which displays the data of the recorded transaction. While the precise information it contains is specific to the command, you can always double-click on the command name to launch its interface, which you can then edit to your particular testing needs. A command interface is displayed hierarchically in its own structure editor within the main eCATT script editor. (For more details on command interfaces, see my previous article in the January/February 2003 issue of this publication.)

- **Offset and length is used to address parts of the date value:** If you used character values in CATT to manipulate dates, you could address sections of the date using offset (which defines the position at which the address begins) and length (which defines the number of characters to be addressed). So, for example, to address the "day" section of a value in a US date field — e.g., *06202003* — you could use offset +2 and length 2, in which case the +2 offset would begin the address after "06" and length 2 would continue it for the next 2 characters ("20"), which represent the 20th day. In eCATT, however, since the date will be rendered in the internal format *20030620*, offset +2 and length 2 will address the last two digits of the year ("03") rather than the day ("20").

If you have stored dates in character values in CATT, after the migration adjust the resulting eCATT scripts as follows:

- Change the data type of the field to ABAP type D or use the dictionary reference SYDATUM.
- If you have programmed the dates using offset and length, try using date arithmetic or a suitable function module to achieve the same results. If you find that you have to use offset and length, reset the values to take into account that the date fields will be rendered in internal format (e.g., in the date example above, use offset +6 and length 2 to address the day).

- *SETVAR*, which was used in CATT to define the values to be used for a test, has been eliminated, so that only an equal sign (=) is required to implement the command. Note also that in eCATT the parameter is preceded by an "E" (denoting that it's an export parameter) instead of an ampersand (&).

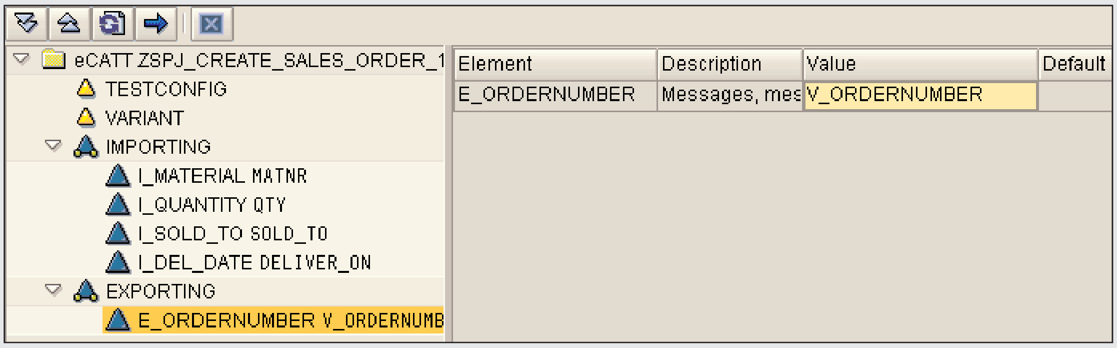
There are two final tasks we need to perform to complete the migration of the *VA01* test module.

First, we must re-create the transaction recording so that it runs against the new CRM-based transaction instead of the old R/3 transaction. The eCATT recording procedure is essentially the same as the CATT procedure, so I won't belabor the details here (see my previous article in the January/February 2003 issue for a more detailed discussion of the tasks involved).

Second, we have to adjust the script's target system definition — i.e., where the transaction recording will run (refer back to the *Target System* field in Figure 7). Remember that by default, the migration tool defines the script's source system (the system from which it was migrated) as the script's target system (the system on which it should now run). In other words, if we were to run the migrated Create Sales Order script right now, it would physically execute on eCATT's 6.20 box, but would log in to the R/3 system and remotely run transaction *VA01* there. Since we want the script to run the transaction on the CRM box, we must change the target system in eCATT by modifying the value of the *Target System* field on the script's attributes maintenance screen — in the example, we would simply change it from *R3* to *CRM*.

At this point, we've successfully migrated our Create Sales Order (*VA01*) module to eCATT and updated it to run against our CRM system. As we are leaving the Create Delivery (*VL01*) module in the R/3 4.6C system and will run it remotely from eCATT, all that's left is to migrate our master CATT procedure so that it can call the two test modules in sequence (remember that you can run a CATT object from eCATT but not the other way around, so the master CATT procedure would not be able to run our newly migrated *VA01* module). But before we do that, you first need to understand the eCATT commands we will need to use to invoke the subordinate modules from the master test script: *REF*, *REFCATT*, and *REMOTECATT*.

Figure 10 The REF Command Interface



eCATT ZSPJ_CREATE_SALES_ORDER_1
TESTCONFIG
VARIANT
IMPORTING
I_MATERIAL MATNR
I_QUANTITY QTY
I_SOLD_TO SOLD_TO
I_DEL_DATE DELIVER_ON
EXPORTING
E_ORDERNUMBER V_ORDERNUMB

Element	Description	Value	Default
E_ORDERNUMBER	Messages, mes	V_ORDERNUMBER	

- **The *REF* command:** In eCATT, the *REF* command serves a purpose similar to the one it served in CATT: it allows you to call one script from another. The *REF* command can only call other eCATT scripts from an eCATT script, however — you cannot use it to call CATT objects from eCATT, or eCATT scripts from CATT. The command has the following syntax:

```
REF ( <script>, <command
  interface>, [<default
  system>]).
```

The default system specified in the *REF* command is drawn from the script's associated system data container. If a command in the script does not have an explicitly assigned target system, eCATT will execute it in the specified default system.

Figure 10 shows what a *REF* command interface would look like in our example scenario. From our "master script," which controls the flow of Company X's order-to-cash process test, we can use the *REF* command to call constituent eCATT scripts — for example, the one that runs the Create Sales Order transaction. On the left side of Figure 10 are the importing and exporting parameters of the called script, to which values are assigned using parameters of the calling script.

- **The *REFCATT* and *REMOTECATT* commands:** New with eCATT are the *REFCATT* and *REMOTECATT* commands, which you can use to call a CATT object from an eCATT script.

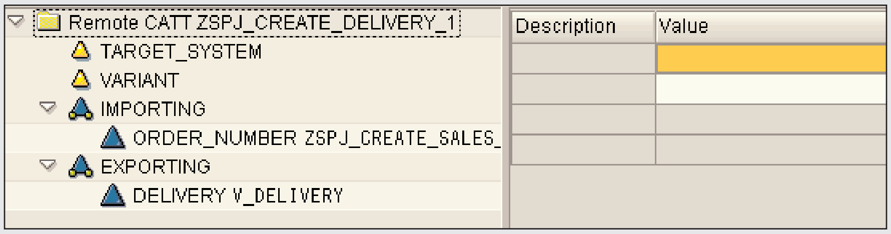
✓ ***Tip***

To call a CATT object residing in the local system, use REFCATT. To call a CATT object residing in a remote system, use REMOTECATT.

REFCATT allows you to call a CATT object residing locally in your central eCATT test system, which would likely be the case if, in the process of consolidating your test resources, you were to transport all of your CATT objects into your central test system. In our example, however, we have a CATT module (Create Delivery) that resides in the R/3 system, which means that we cannot use the *REFCATT* command.

The *REMOTECATT* command, on the other hand, *can* call a CATT object residing in a different system. For our example scenario, the command syntax would be:

Figure 11 *The REMOTECATT Command Interface*


Remote CATT ZSPJ_CREATE_DELIVERY_1
TARGET_SYSTEM
VARIANT
IMPORTING
ORDER_NUMBER ZSPJ_CREATE_SALES_
EXPORTING
DELIVERY V_DELIVERY

Description	Value

```
REMOTECATT ( ZSPJ_CREATE_DELIVERY,
  ZSPJ_CREATE_DELIVERY_1, R3).
```

The *REMOTECATT* command interface (*ZSPJ_CREATE_DELIVERY_1*) for our example would look like **Figure 11**.

✓ ***Tip***

Using the TARGET_SYSTEM field in the REMOTECATT command interface, you can specify that the remote CATT object (Create Delivery in our example) be executed in a third system (i.e., instead of in the system in which it resides or in the eCATT system). This does not apply in our example, since we want the VL01 module to run in its R/3 system; however, if you used a central test system for CATT, but are not using it for eCATT because it is running a release lower than 6.20, you might want to take advantage of this option. The entry in the TARGET_SYSTEM field must be an entry specified in the associated system data container, but the RFC destination assigned to the logical system in the system data container does not have to be declared in the central test system. Instead, it must be declared in the system in which the CATT procedure resides.

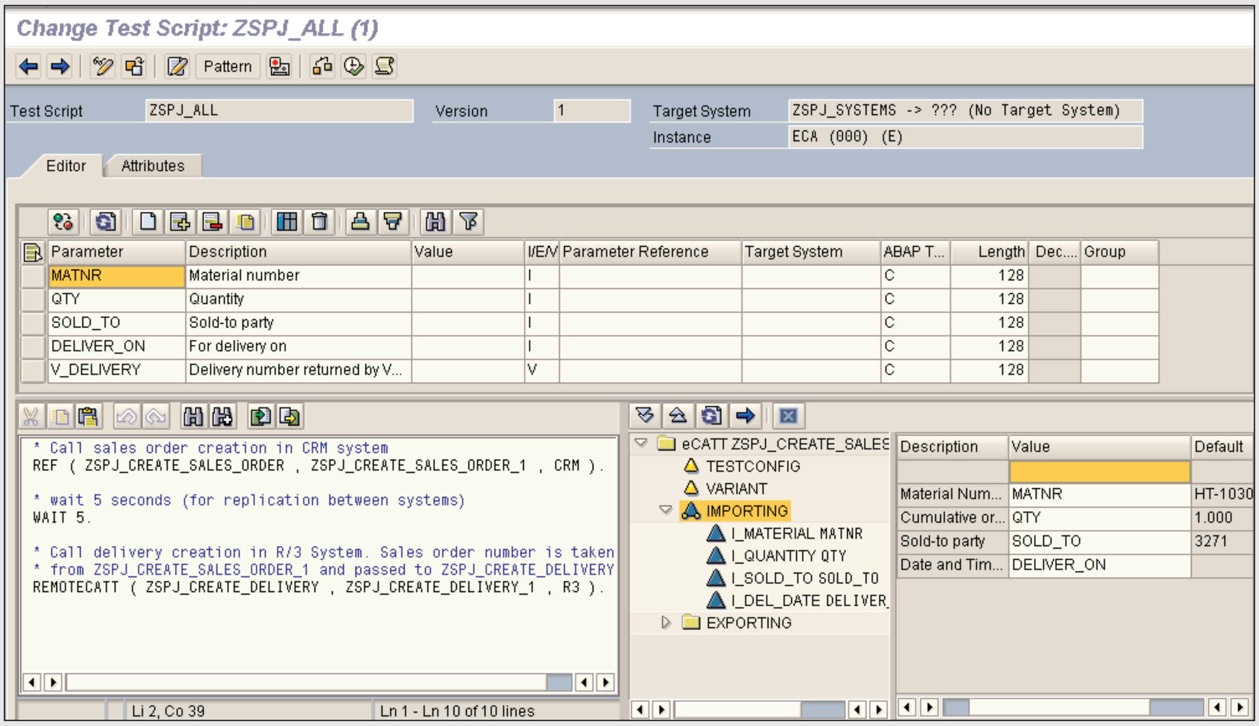
As you can see in Figure 11, the *REMOTECATT* command has importing and exporting parameters similar to *REF* (though not shown, the same is true for *REFCATT*). In our example, the *REMOTECATT* command imports the relevant order number (which comes from the preceding *REF* command) into the Create Delivery module and then exports the delivery number created by the remote module to the parameter *V_DELIVERY* in the calling script.

Migrate the CATT Master Test Procedure

With the *REF*, *REFCATT*, and *REMOTECATT* commands in our mental toolbox, we're now ready to migrate our example master test procedure from CATT to eCATT. The actual process parallels the one used previously for migrating the Create Sales Order module, so let's assume that we've already run the migration tool successfully and that the CATT procedure is now an eCATT test script.

Recall the earlier discussion of the benefits and drawbacks of migrating scripts versus reimplementing them entirely. While it would make sense to reimplement a script for situations where the code is entirely unusable (because the transaction in question is no longer used, for example), it is generally a good idea to migrate whenever possible, as we did in the example scenario here. Overall, the migration approach saved us time — the migration tool automatically created the eCATT script for our master CATT procedure and brought across its administrative attributes,

Figure 12   The Completed eCATT Master Script



Test Script	ZSPJ_ALL
Version	1
Target System	ZSPJ_SYSTEMS -> ??? (No Target System)
Instance	ECA (000) (E)

Parameter	Description	Value	I/E/V	Parameter Reference	Target System	ABAP T...	Length	Dec....	Group
MATNR	Material number		I			C	128		
QTY	Quantity		I			C	128		
SOLD_TO	Sold-to party		I			C	128		
DELIVER_ON	For delivery on		I			C	128		
V_DELIVERY	Delivery number returned by V...		V			C	128		

```
* Call sales order creation in CRM system
REF ( ZSPJ_CREATE_SALES_ORDER , ZSPJ_CREATE_SALES_ORDER_1 , CRM ).

* wait 5 seconds (for replication between systems)
WAIT 5.

* Call delivery creation in R/3 System. Sales order number is taken
* from ZSPJ_CREATE_SALES_ORDER_1 and passed to ZSPJ_CREATE_DELIVERY
REMOTECATT ( ZSPJ_CREATE_DELIVERY , ZSPJ_CREATE_DELIVERY_1 , R3 ).
```

Li 2, Co 39   Ln 1 - Ln 10 of 10 lines

```
eCATT ZSPJ_CREATE_SALES
  TESTCONFIG
  VARIANT
  IMPORTING
    I_MATERIAL MATNR
    I_QUANTITY QTY
    I_SOLD_TO SOLD_TO
    I_DEL_DATE DELIVER
  EXPORTING
```

Description	Value	Default
Material Num...	MATNR	HT-1030
Cumulative or...	QTY	1.000
Sold-to party	SOLD_TO	3271
Date and Tim...	DELIVER_ON	

parameters, and variables. All that is left for us to do is to remove the target system specification (since we want the master script to execute locally on the central 6.20 test system, and not in R/3)[15] and add the appropriate commands to call the local eCATT Create Sales Order script and the remote R/3 Create Delivery module (using the *REF* and *REMOTECATT* commands, respectively). The end result is shown in **Figure 12**.

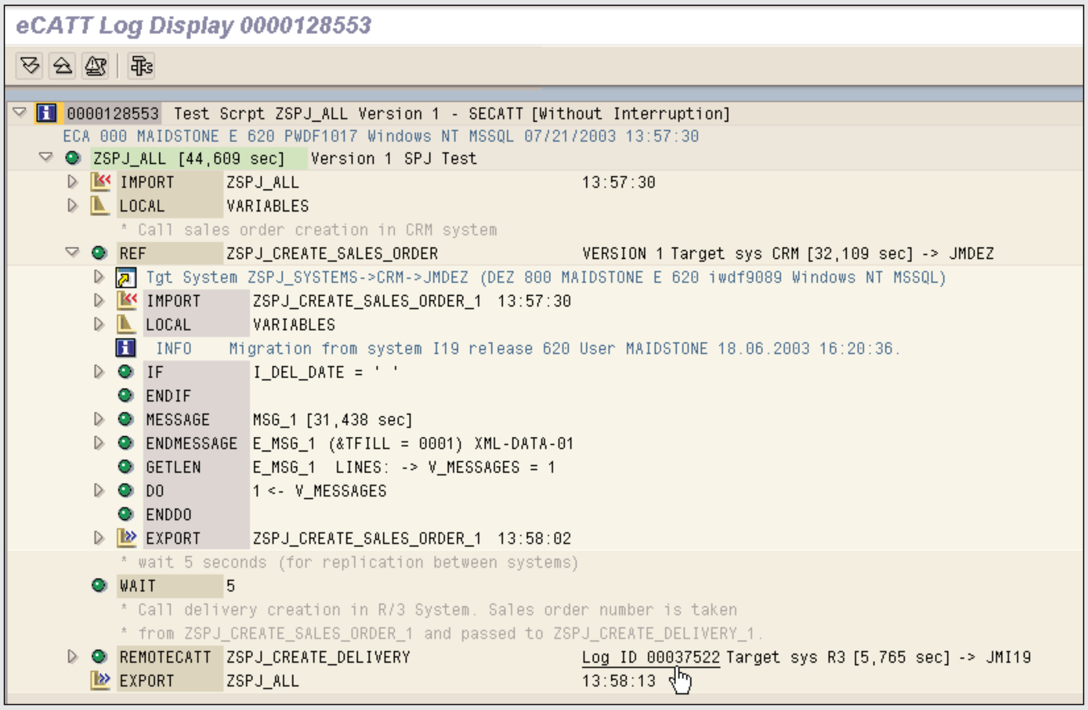
If you take a closer look at the code in Figure 12, you will see that we're first calling our local eCATT script *ZSPJ_CREATE_SALES_ORDER_1*, but passing to it the name of the system on which it should run (*CRM*). Also notice that we're telling eCATT to connect to R/3 and call the *ZSPJ_CREATE_DELIVERY* module residing there.

Running the example test script produces the log shown in **Figure 13**. Notice how the log includes a high level of detail for the local scripts called using the *REF* command, but only a single line (a link to the CATT log in the remote system) for those called with *REMOTECATT*. You can click on the link (the *Log ID* entry at the bottom of the screen) to view the remote log.

Now that you've got a solid understanding of the fundamentals of migrating CATT objects to eCATT, let's look at some of the more advanced techniques and considerations involved in performing these migrations.

[15] Remember that by default, the migration tool defines (on the general attributes screen) the system the script was migrated from as the script's target system. Here we want the script to run locally, so it should not be assigned a target system.

Figure 13 **The Log Produced by Our eCATT Master Script**



```
eCATT Log Display 0000128553

0000128553 Test Scrpt ZSPJ_ALL Version 1 - SECATT [Without Interruption]
  ECA 000 MAIDSTONE E 620 PWDF1017 Windows NT MSSQL 07/21/2003 13:57:30
  ZSPJ_ALL [44,609 sec]  Version 1 SPJ Test
    IMPORT      ZSPJ_ALL                              13:57:30
    LOCAL       VARIABLES
      * Call sales order creation in CRM system
    REF         ZSPJ_CREATE_SALES_ORDER               VERSION 1 Target sys CRM [32,109 sec] -> JMDEZ
      Tgt System ZSPJ_SYSTEMS->CRM->JMDEZ (DEZ 800 MAIDSTONE E 620 iwdf9089 Windows NT MSSQL)
      IMPORT      ZSPJ_CREATE_SALES_ORDER_1  13:57:30
      LOCAL       VARIABLES
       INFO     Migration from system I19 release 620 User MAIDSTONE 18.06.2003 16:20:36.
      IF          I_DEL_DATE = ' '
      ENDIF
      MESSAGE     MSG_1 [31,438 sec]
      ENDMESSAGE  E_MSG_1  (&TFILL = 0001) XML-DATA-01
      GETLEN      E_MSG_1  LINES: -> V_MESSAGES = 1
      DO          1 <- V_MESSAGES
      ENDDO
      EXPORT      ZSPJ_CREATE_SALES_ORDER_1  13:58:02
      * wait 5 seconds (for replication between systems)
    WAIT        5
      * Call delivery creation in R/3 System. Sales order number is taken
      * from ZSPJ_CREATE_SALES_ORDER_1 and passed to ZSPJ_CREATE_DELIVERY_1.
    REMOTECATT  ZSPJ_CREATE_DELIVERY                  Log ID 00037522 Target sys R3 [5,765 sec] -> JMI19
    EXPORT      ZSPJ_ALL                              13:58:13
```

Recursive Migration: A Shortcut for Migrating Multiple CATT Objects to eCATT

The approach we've taken in our example scenario is to migrate each CATT object individually and then reassemble the call chains in eCATT using the *REF* and *REMOTECATT* commands. While this was useful in order to clearly demonstrate the migration concept, if you have a lot of CATT resources to migrate (which in a real-world environment is most likely to be the case), this one-by-one approach can take a lot of time. Is there a quicker way? Thankfully, there is! It's called *recursive migration*.

Remember Figure 1, which showed Company X's test for its order-to-cash business process — a single master CATT procedure that called two modules for testing individual applications (the R/3 transactions Create Sales Order and Create Delivery)?

Well, rather than migrate the Create Sales Order test module and the master CATT procedure separately, as we did in the previous sections, we could have just specified the master procedure for migration and checked the *Recursive* checkbox in Figure 7. The migration tool would have then automatically migrated the master procedure plus *both* of the called modules in one step!

You could then use the migrated Create Delivery test module to test the R/3 Create Delivery transaction by running it in eCATT, but against the R/3 system using the *REFCATT* command.

Figure 14 *Recursive and Non-Recursive Migration*

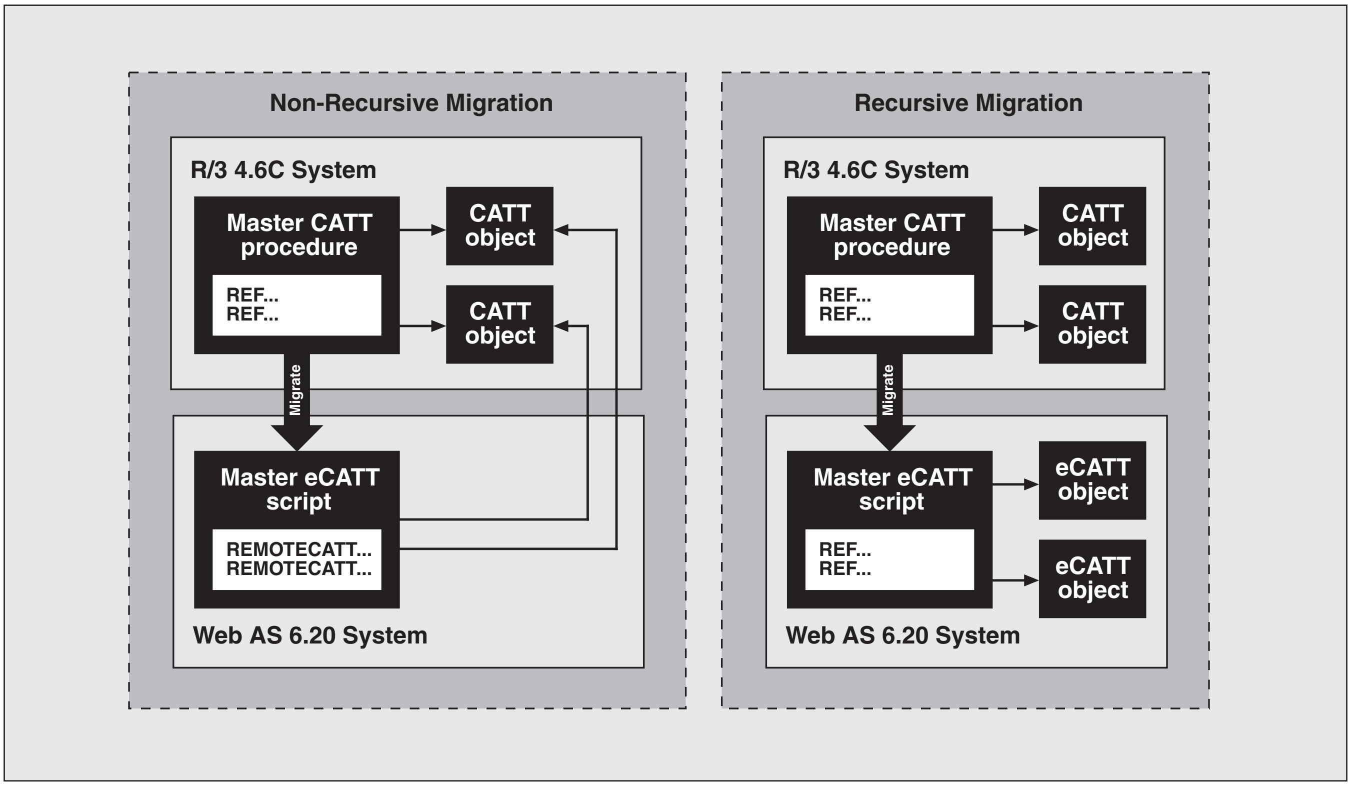


Figure 14 shows a visual comparison of non-recursive (i.e., one object at a time) and recursive (i.e., multiple objects at the same time) migration.

Another key advantage of recursive migration is that eCATT automatically updates all of the test object references in the new scripts. If you migrate the top-level CATT procedure, but none of the called CATT objects, you are left with a raft of *REMOTECATT* references, which is fine as long as those called objects are still running remotely under CATT.

However, once the called objects are migrated, and thus local to the master eCATT script, their *REMOTECATT* references need to become *REF* commands, and it is impossible to determine automatically which ones need to be changed. Consequently, I recommend that you use the recursive migration option to save yourself extra work further on down the line.

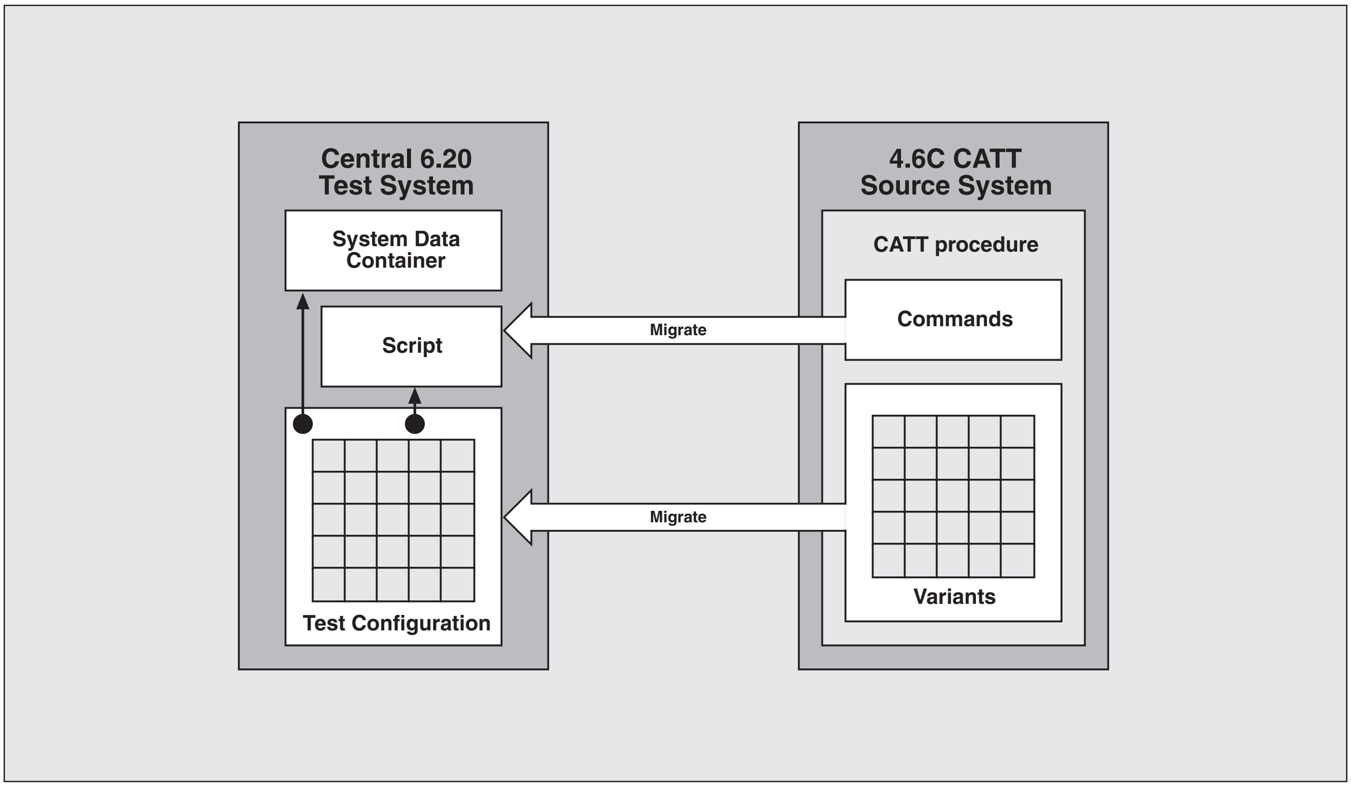
✓ ***Tip***

eCATT's automatic conversion of REMOTECATT commands to REF commands (during recursive migration) will save you a lot of work if you plan on migrating most of your subordinate CATT objects as well.

Migrating CATT Procedures with Variants

One of the issues involved in migrating test objects from CATT to eCATT is that CATT and eCATT sometimes have different requirements as to how test resources are arranged and managed. We have already

Figure 15 *An Example Migration of a CATT Procedure with Variants*



seen one of these differences (you can store a CATT procedure in any system, but eCATT scripts must be stored in the central test system). Another is that CATT allows you to store different versions of the test data — in the form of variants — as part of a test procedure, while in eCATT all test data is stored in test configurations.

In practice, this means that CATT procedures containing variants cannot simply be migrated to eCATT scripts. Instead, they are migrated in a process that splits up the test instructions and test data according to eCATT principles (i.e., into test data containers, test scripts, and system data containers) before reuniting them in test configurations to execute the test (refer to the sidebar on page 4 about the new eCATT model). As with any other migration, the new eCATT scripts are created in the central test system regardless of where the CATT procedures previously resided.

✓ ***Tip***

If you previously used external variants with CATT, you must upload them into their corresponding CATT procedures before you migrate them.

Figure 15 shows how this process works. First, the CATT commands are migrated to a new eCATT script, but without the variants. Then, eCATT creates a test configuration with the same name as the new test script and places the CATT variants into it. Finally, the script is linked to the test configuration, as is the system data container used to originally specify the source system of the original CATT procedure.

This difference in how variants are handled is reflected in the parameter lists of the *REF*, *REFCATT*, and *REMOTECATT* commands. When you call a CATT procedure remotely using *REFCATT* or *REMOTECATT*, you can specify the name of a variant (in the *VARIANT* parameter) with which eCATT should populate the procedure's import parameters. In contrast, the *REF* command (used to call other eCATT scripts from eCATT) requires not only a variant name, but also the name of the test configuration in which it is stored.

Handling Obsolete CATT Scripting Features

So far, we've seen that the vast majority of CATT objects can be ported automatically and painlessly to eCATT using eCATT's built-in migration tool. There are a few CATT scripting features, however, that have been dropped or replaced in eCATT. Fortunately, the eCATT migration tool helps you spot where these features are used by inserting comments into the script. Here we'll look at two of the most significant differences between CATT and eCATT scripting features.

✓ ***Tip***

A list of common CATT commands and their eCATT equivalents and a list of select CATT variables that have changed names in eCATT are available for download at ***www.SAPpro.com****.*

Difference #1: Support for the &LDS Variable

In CATT, you could address a remote system in one of the following ways:

- Use the set/get parameter *RFC_ID*.
- Assign an RFC destination explicitly when you start a CATT procedure.
- Set the CATT variable *&LDS* within the CATT procedure itself.

If you use either of the first two options, you could set a single destination that would be valid for the entire procedure. By using the *&LDS* variable within the procedure, you could switch destinations during the procedure itself.

The *&LDS* variable does not work with eCATT because *&LDS* destination assignment is global — you set the destination, which then applies to all subsequent commands until you revoke it. eCATT, on the other hand, supports only explicit destination assignment (i.e., logical system names stored in system data containers) — the majority of eCATT commands can take a specific target system as one of their arguments, making eCATT's approach far less error-prone and more flexible.

Therefore, the eCATT migration tool will flag legacy statements like

```
SETVAR &LDS = 'MY_RFC_DEST'.
```

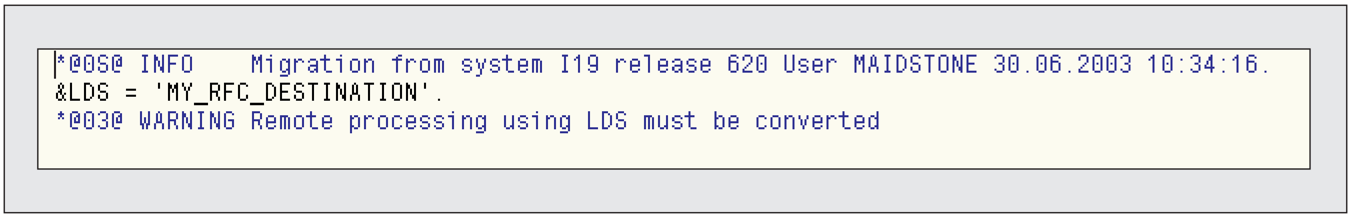
with a message informing you that the migration contains errors (see **Figure 16**).

To mitigate such an error, you must create a system data container (or extend an existing one) with entries for all of the RFC destinations you'll need. Then you must remove all *&LDS* statements and instead specify the target system (using the new logical name) explicitly in each command you want to execute remotely. Most commands, like *REFCATT* and *REMOTECATT*, accept this logical system name as the last argument in their parameter list, as the code excerpt in **Figure 17** shows.

✓ ***Tip***

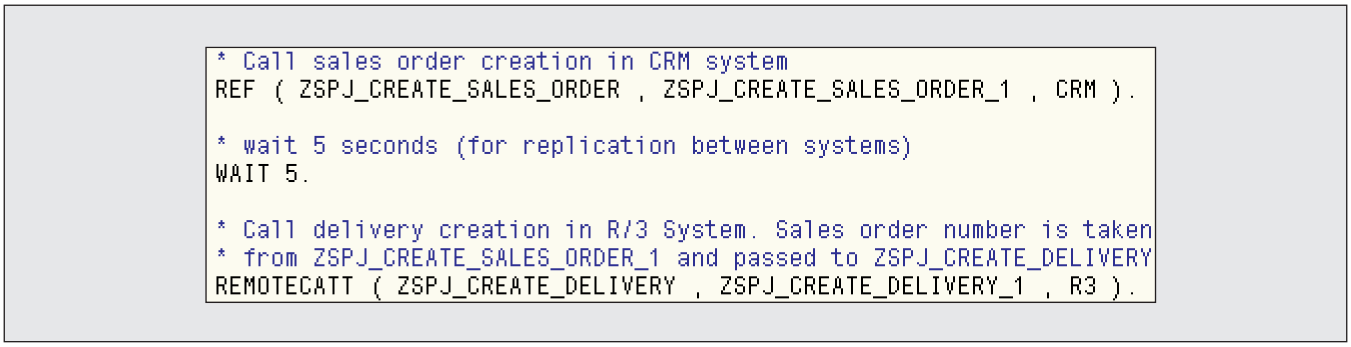
If it turns out that most or all of the commands target a single system, you can declare this as the target system at script level (by entering it in the script attributes). This saves you from having to enter the name in every single command.

Figure 16 Error Message After a Migration Involving the &LDS Variable



```
*@0S@ INFO    Migration from system I19 release 620 User MAIDSTONE 30.06.2003 10:34:16.
&LDS = 'MY_RFC_DESTINATION'.
*@03@ WARNING Remote processing using LDS must be converted
```

Figure 17 Code from Figure 12

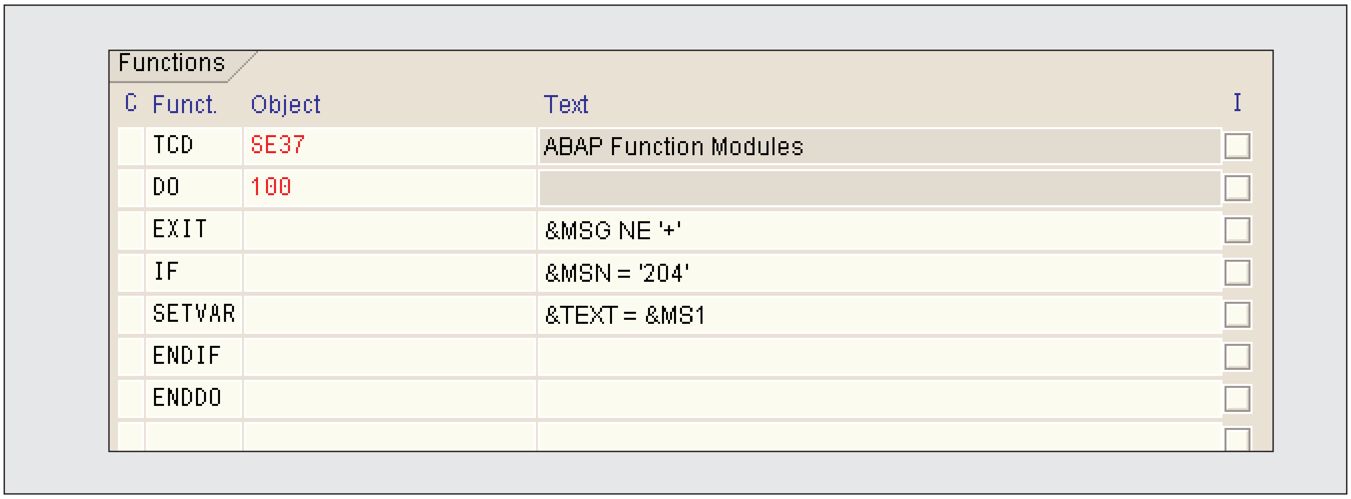


```
* Call sales order creation in CRM system
REF ( ZSPJ_CREATE_SALES_ORDER , ZSPJ_CREATE_SALES_ORDER_1 , CRM ).

* wait 5 seconds (for replication between systems)
WAIT 5.

* Call delivery creation in R/3 System. Sales order number is taken
* from ZSPJ_CREATE_SALES_ORDER_1 and passed to ZSPJ_CREATE_DELIVERY
REMOTECATT ( ZSPJ_CREATE_DELIVERY , ZSPJ_CREATE_DELIVERY_1 , R3 ).
```

Figure 18 CATT Technique for Catching a Specific Result Message



Functions

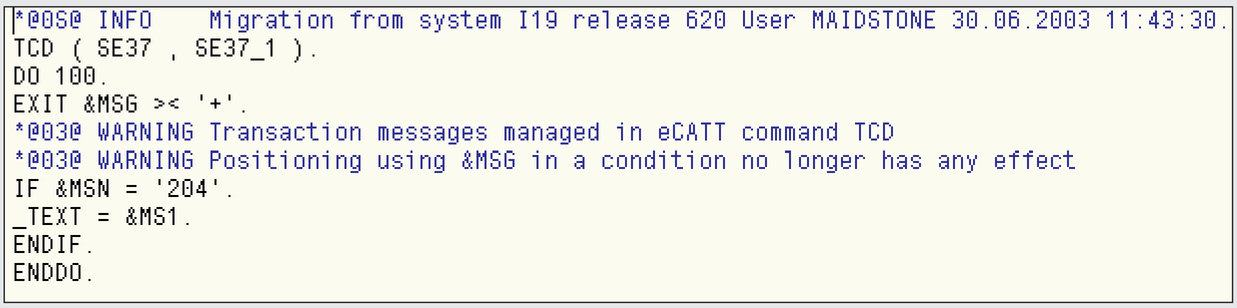
C	Funct.	Object	Text	I
	TCD	SE37	ABAP Function Modules	☐
	DO	100		☐
	EXIT		&MSG NE '+'	☐
	IF		&MSN = '204'	☐
	SETVAR		&TEXT = &MS1	☐
	ENDIF			☐
	ENDDO			☐
				☐

Difference #2: Accessing Result Messages

The other area in which CATT and eCATT scripting features differ significantly is the way in which you loop through all of the system messages that occurred during a transaction test and extract their variable parts. In CATT, you could do this using a conditional loop against the message object *&MSG*. A set of special variables then allowed you to access particular elements of each message, such as its type, class, number, and the variables it contained.

Figure 18 shows an example of this technique. Here, the script tries to locate message *204* within the set of issued messages and place the contents of its first variable part into a parameter in the CATT procedure.

Figure 19 *Figure 18's Code Immediately After Migration to eCATT*



```
*@0S@ INFO    Migration from system I19 release 620 User MAIDSTONE 30.06.2003 11:43:30.
TCD ( SE37 , SE37_1 ).
DO 100.
EXIT &MSG >< '+'.
*@03@ WARNING Transaction messages managed in eCATT command TCD
*@03@ WARNING Positioning using &MSG in a condition no longer has any effect
IF &MSN = '204'.
_TEXT = &MS1.
ENDIF.
ENDDO.
```

Now look at **Figure 19**. This is the script output by eCATT's migration tool. Notice the "warning" comments that alert you to an unsupported feature. While the messages can be hard to understand, they are essentially telling you that the existing looping approach of using "+" and "-" is no longer supported.

To migrate this code successfully to eCATT, you need to know about two of eCATT's new features. First, eCATT provides a new variable, *&MSX*, that contains the total number of messages sent by the last transaction executed. You use this variable to detect when your loop should exit. Second, the TCD command now automatically loads all messages into an internal table called *MSG*. You use this table to access one or all of the messages and their variables using the following syntax:

```
<command_interface>-MSG[<line
  index>]-component.
```

After implementing these changes, the code in Figure 19 looks like this:

```
DO &MSX.
IF ( SE37_1-MSG[&LPC]-MSGNR =
  '204' ).
_TEXT = SE37_1-MSG[&LPC]-MSGV1.
ENDIF.
ENDDO.
```

This eCATT code works just like the original CATT code shown in Figure 18, but it is much easier to read and understand. And while the process may seem time-consuming, once you convert one or two, the rest will take you only a few minutes to complete.

Conclusion

With its ability to test controls-based, distributed, and web applications, eCATT is a significant step forward from its predecessor. Such advances often require significant reengineering of existing technical assets or reimplementing them altogether. With eCATT, this is not the case! This article has given you a jump-start on using eCATT's robust migration tool to port your CATT objects to the new tool with little or no redevelopment required. You've also seen how you can migrate just your master test procedures and remotely call CATT objects (on 4.6C+ Basis systems). With this knowledge, you can assess your own current system landscape and feel confident forming a strategy for your own migration to eCATT.

Jonathan Maidstone graduated from the University of Bristol (UK) in 1996 with a degree in modern languages and joined SAP as a translator for the ABAP Language and ABAP Workbench departments. After a spell as a technical writer, during which he was involved with the SAP Control Framework and DCOM Connector projects, he is now a product management specialist with particular responsibility for rolling out eCATT, SAP's new test tool. Jon can be contacted at jonathan.maidstone@sap.com.