SAPinsider

This article appeared in the Apr▪May▪Jun 2007 issue of *SAP Insider* and appears here with permission from the publisher, Wellesley Information Services (WIS), **www.WISpubs.com**.

WIS

Say Goodbye to Manual Performance Tests

A How-To Guide for Automating Performance Testing Using ST30

Identifying and solving performance problems in your coding — the additional programs that your custom development teams are writing in your SAP systems — *before* they slow down mission-critical business processes is a must. Otherwise, you risk longer response times, higher TCO, and possible production interruptions, slowdowns, or standstills. Yet, to date, many performance tests still have to be done manually — and on each system individually — even in highly complex system landscapes.

To move away from time-consuming manual testing, development teams are well advised to embed *automated* testing procedures into their custom coding processes. This can best be achieved by submitting custom coding to an automated performance test, like the one provided by SAP's **Global Performance Analysis** tool (transaction ST30). You may be surprised to learn that *your SAP system already includes this tool*; all you need to do is set it up and use it, a task that those familiar with performance testing can do in, let's say, less than thirty minutes.

In an earlier Performance & Data Management Corner column,[1] we introduced the principles and benefits of **automated performance testing and analysis**[2] in an SAP environment (see first sidebar on the next page). We looked at the entire automated performance testing process, the key requirements for automated tests, and a checklist for identifying the culprits of poor performance. Now, you'll learn how to actually *perform* these automated performance tests with transaction ST30.

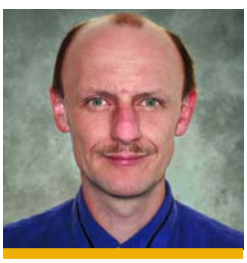
Conduct Automated Peformance Tests with ST30 — A Step-by-Step Overview

The Global Performance Analysis tool (transaction ST30) enables you to automate the performance testing of business scenarios in complex SAP system landscapes, delivering performance results for all — even unknown — ABAP and non-ABAP system components (accessible via RFC destinations maintained in transaction SM59) that are involved in a specific business process.

ST30 collects relevant performance data across system boundaries and keeps it on a central database, enabling you to compare results or perform statistical evaluations — for regression testing, for example.

So how does it work? Global Performance Analysis invokes eCATT[3] scripts from a central test system.[4] These scripts run recorded scenarios, involving all of the relevant components of the system landscape under test. ST30 retrieves the performance figures generated from these runs by:

- Using statistical records (transactions STAD, ST03N, ST03G, etc.) as well as SQL trace data provided by transaction ST05
- Integrating Code Inspector (transaction SCI) functionality



Armin Hechler-Stark studied computer science and electrical engineering at the University of Saarbrücken. He joined SAP in 1994 as an application developer for master data management. He has been a member of SAP's Performance, Data Management, and Scalability team since 2000, and is currently responsible for developing automated performance measurement tools. He can be reached at armin.hechler-stark@sap.com.

Helmut Stefani studied applied linguistics at the Johannes Gutenberg University in Mainz, where he graduated as a technical translator for English and Spanish. He has been a member of SAP's Performance, Data Management, and Scalability team since 1997, working on the documentation, product management, and rollout of data archiving topics. Helmut has authored several publications on data management and data archiving. He can be reached at helmut.stefani@sap.com.

[1] See "Lower TCO and a Better End-User Experience Through Automated Performance Testing and Analysis" by Armin Hechler-Stark in the April-June 2006 issue of *SAP Insider* (www.SAPinsideronline.com).

[2] The goal of *performance testing* is to find out where program performance is less than optimal; that of *analysis* is to remedy these situations by interpreting performance test results. Whenever the term "performance testing" is used in this article, it refers by extension to "performance analysis."

[3] SAP's extended Computer Aided Test Tool (eCATT), transaction SECATT, is based on test scripts that contain all of the user interaction steps of the scenario being tested. eCATT scripts, once recorded or written, can be modified by manual editing at any time. eCATT functionality is available as of SAP Web Application Server (SAP Web AS) 6.20, but the systems under test can be any systems as of release SAP R/3 4.6C.

[4] The *central test system* is the system in which transaction ST30 runs. This system is different from the *system* (or systems) *under test*.

Automated Performance Testing and Its Benefits

Performance tests at the application level help you to quickly detect excessive resource consumption that could lead to degraded performance across the entire system landscape. Automating these performance tests involves automatically and repeatedly starting a particular business process or scenario, involving the appropriate components in the system landscape, and collecting and centrally storing relevant performance data. The advantages of automated performance tests are numerous; here are the most important ones:

- You can reproduce tests efficiently and retrieve and reuse past test results (performance figures) at any time.
- You can avoid costly manual performance tests.
- You can run test series to monitor performance behavior over time.
- You can compare performance tests to each other – even over extended time intervals – to detect possible performance degradation.

Before You Get Started with ST30

A Prerequisite Checklist

- ☑ Have a general knowledge – an overview is sufficient – of SAP performance testing tools, including statistical records, SQL trace, and Code Inspector.
- ☑ Be familiar with eCATTs (transaction SECATT).
- ☑ Make test scenarios available as an eCATT test configuration in the central test system.
- ☑ Enable the systems under test to work with eCATTs and GUI scripting.[5] In your local GUI, activate the option *Enable Scripting*, but deactivate both *Notify when...* indicators in the Options menu (*Alt F12* → *Options* → *Scripting* tab).
- ☑ Make the required profiles and authorizations available in the systems to be tested *and* in the central test system.
- ☑ Make sure you define the RFC destinations where eCATT test configurations are supposed to run *without load balancing* (see transaction SM59), since the performance figures created in a system that runs on multiple servers are server-specific.
- ☑ Execute the eCATT test configurations repeatedly from within ST30 to ensure more reliable results. While doing so, take special care that each test run does not influence subsequent runs. For example, the test data, such as database data, must not be changed between individual runs. Only with this consistency can you be sure that the test conditions are reliable and identical for every test run.

[5] GUI scripting is a technology that enables tests to be initiated from outside the part of the system landscape being tested. This ensures that there's no performance-relevant influence on the tested system components so you can obtain highly stable and reliable results.

ST30 brings all of these standalone, manual SAP performance testing tools into one central, automated tool; essentially, it does all the work for you, so you no longer have to manually log in to every system component and test each one individually.

Getting started with automated performance testing involves three major steps:

1. Set up eCATT test configurations for use in ST30
2. Use eCATT test configurations from within ST30
3. Display and interpret the test results

Let's break these into more detailed action items.

Step #1: Set Up an eCATT Test Configuration for Use in ST30

Before you conduct an automated performance test with ST30, you will need to set up an eCATT test configuration (containing a recorded test script) that will form the basis of the performance test. Proceed as follows:

1. To accurately measure the systems you want to test, first turn off the Easy Access menu while recording the test script; the menu could affect the performance test results. To stop this menu in the session, enter "/n" once or several times. You'll know that you have successfully stopped the menu if the screen displays the *Start SAP Easy Access* button.

2. Record, with GUI scripting, the test scripts you will use within the test configuration.

3. Maintain your test configuration, which includes your recorded test script, in the central test system.

4. Use a realistic quantity of data to serve as a basis for testing in each system; in other words, make sure that the amount of data the eCATT test configuration uses within the system under test is approximately equal to the quantity that's normally in the live system. You can create this data using, for example, client copy, extra eCATT scripts, or reports to reset the data to a default.

5. To help differentiate between steps that you want to measure and those that you do not – those steps that were performed to ensure identical testing conditions, for example – use data-correcting

reports with self-explanatory names within the eCATT test scripts. For example, you could call a report used within a test script "Z_RESET_MY_DATABASE_TEST_DATA".

6. Before you use the eCATT test configuration in ST30, execute it in transaction SECATT to check that it runs correctly.

Step #2: Use eCATT Test Configurations from Within ST30

Let's now go through the general procedure for conducting automated performance tests. For the full details, please refer to Application Help.[6]

Specify the Run Parameters

When specifying the Run Parameters, the following fields require special attention (see **Figure 1**):

- *No. of eCATT Preprocs* ❶ – eCATT preprocesses precede the main runs from which the system retrieves performance figures. They set the resources used for the run (program buffer, table buffer, and so on) to a well-defined status before the performance of the subsequent eCATT main runs is measured. We recommend that you specify a minimum of five preprocesses to be sure that, for example, the system buffers are filled and that recently modified coding is generated. Specifying only one preprocess isn't sufficient; you may have some underlying side effects that could only be determined with multiple preprocesses.

- *No. of eCATT Main Runs* ❷ – These are the runs you're measuring the performance behavior of, and they are executed after the eCATT preprocesses. We recommend that you specify ten or more main runs. That way, ST30 will automatically present the resulting performance figures as average values derived from the main runs, giving you more accurate, stable, and reliable results.

- *With Run for SQL Trace* ❸ – If you activate this indicator, one additional run of the test configuration is executed to create a SQL trace. The purpose of this additional run is to avoid having the SQL trace influence the main run's performance measurements.

Create a Performance Checklist and Include Code Inspector Results

If you do nothing more than set up the run parameters as defined above, you'll end up with standard performance test results, including statistical and SQL trace data. To obtain more insight, you'll want to dig deeper into the code by including Code Inspector results or creating a performance checklist.

If you activate *Checklist* in the Programming Guidelines (see **Figure 2** on the next page), your finished product will include a list of results in the

ST30 brings standalone, manual SAP performance testing tools into one central, automated tool; essentially, it does all the work for you, so you no longer have to manually log in to every system component and test each one individually.

[6] To access Application Help from within ST30, select *Help → Application Help*. Alternatively, follow this menu path in the SAP Help Portal (http://help.sap.com): *SAP NetWeaver Library → SAP NetWeaver by Key Capability → Solution Life Cycle Management by Key Capability → Testing → Test Workbench → Global Performance Analysis.*

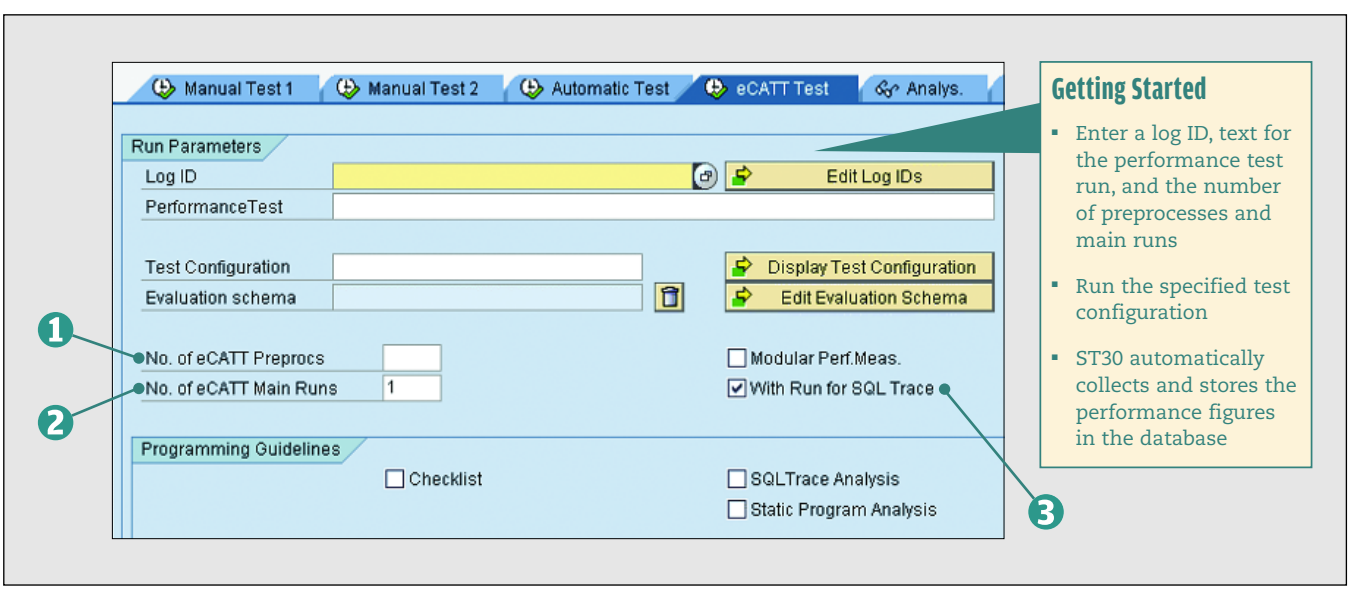


FIGURE 1 ▲ Set the Run Parameters in the ST30 screen

✔ **NOTE!**

The status information within the Programming Guidelines Checklist displayed in the *Appropriate Indexes*, *Complete WHERE clauses*, and *Buffers and Caches* columns, for example, is derived from the Code Inspector. The status information in the other columns is derived from the statistics data and the SQL trace.

Programming Guidelines Checklist (see **Figure 3**). The parameters tested here are benchmark, best-practice values stemming from practical experience. For example, based on these best practices, the response time should be less than two seconds, and the maximum number of round trips per user interaction step should be two. The creation of the checklist is only possible if you also checked *SQL Trace Analysis* (as shown in Figure 2) and *With Run for SQL Trace* (refer back to Figure 1), because some results of the checklist are based on the analysis of that SQL trace. You'll note that some of the result columns are inactive; they are only meant for future use.

If you activate *SQL Trace Analysis*, the Code Inspector will check the SQL trace created during the performance test. The purpose of this function is to identify inefficient database accesses that may arise from dynamic SQL statements in the program under test. For example, the checklist results for *Appropriate Indexes*, *Complete WHERE clauses*, and *Buffers and Caches* are derived from the automatic SQL trace analysis (executed by the Code Inspector).

The *Static Performance Analysis* indicator determines whether the performance checks of the Code Inspector are used with the programs involved during the performance test (eCATT). The purpose of this function is to identify inefficient database accesses also in the static coding.

TIP!

Include Code Inspector results in your performance test if you discover any abnormalities in a standard test run; you'll be able to investigate what went wrong in more detail.

Specify the Central Monitoring System

If your system landscape includes non-ABAP (e.g. Java) component types, you'll need to include distributed statistical records (DSRs; see sidebar on the next page) in your performance test. To do this, specify the central monitoring system and activate the indicator *With Distributed Statistics Data* (see **Figure 4**).

Run the Performance Test

Once you have specified all applicable entries, start the measurement runs with *eCATT Test Only*. After the runs are complete, check for possible error messages in the tab *Autom. Test: Logb*. For more information on how to use this function, see Application Help.

Compare Performance Test Data

To compare the data of different performance tests, you must specify the required information in the Data Comparison section of the ST30 screen (see **Figure 5**).

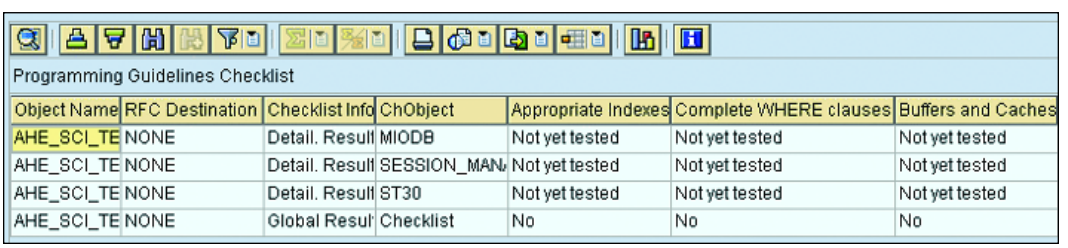
Step #3: Display the Performance Test and Analysis Results

Once your performance tests are complete, you'll want to review the results to analyze improvements or to determine any corrective steps you'll want to take next.

FIGURE 2 ▲ Create a performance checklist (optional) and activate Code Inspector functions

✔ **NOTE!**

The only way to analyze the performance of non-ABAP (e.g., Java) components is to include DSRs in your test. If you are running only ABAP systems within your SAP system landscape, however, you do not need to specify the central monitoring system.



Object Name	RFC Destination	Checklist Info	ChObject	Appropriate Indexes	Complete WHERE clauses	Buffers and Caches
AHE_SCI_TE	NONE	Detail. Result	MIODB	Not yet tested	Not yet tested	Not yet tested
AHE_SCI_TE	NONE	Detail. Result	SESSION_MAN	Not yet tested	Not yet tested	Not yet tested
AHE_SCI_TE	NONE	Detail. Result	ST30	Not yet tested	Not yet tested	Not yet tested
AHE_SCI_TE	NONE	Global Result	Checklist	No	No	No

FIGURE 3 ▲ The result: a checklist showing how your coding complies with defined programming guidelines and how its performance complies with defined best-practice values

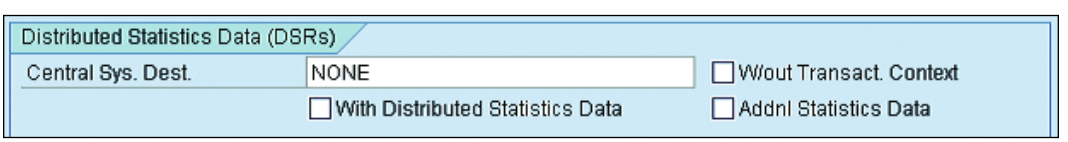


FIGURE 4 ▲ Specify the central monitoring system if your landscape includes non-ABAP components and you want your test to read distributed statistical records (DSRs)

View the Test Results

You can view ST30 test results (only from the run that has just been executed) in the *Data Overview* tab. The information displayed on this tab is identical to that provided in the *Aggregated Performance Data for Each Performance Test* view in transaction ST33. In addition, ST33 displays other performance figures in a raw (non-aggregated) format, as well the data from all previous performance tests.

In ST30's Analysis Results section, you can see the results obtained from Code Inspector checks and the results derived from statistical data and trace data. This tab consists of two main sections displaying different results: Analysis Results (see **Figure 6**) and Programming Guidelines Checklist (refer back to Figure 3).

Each Analysis Results line provides only a brief overview of the test results, including how many errors were found in the executed SQL statements. To obtain more detailed information, select a line and choose *SQL Trace Analysis* or *Stat. Program Analysis*. This will take you directly to the respective Code Inspector screen in the system in which the performance test was running.

The Code Inspector even allows you to drill down to the level of individual messages and, if applicable, to the line in the coding where the performance

The Inner Workings of ST30

Tools and Functions Integrated in SAP's Global Performance Analysis

Apart from the eCATT function (transaction SECATT), the following functions are also integrated in ST30:

☑ Central Monitoring System and DSRs

An hourly collector process requests the distributed statistical records (DSRs) from the agents of actively monitored system components and aggregates this data. Transaction ST03G displays the current DSRs of the monitored system landscape. DSRs can be provided by every component type (including ABAP types such as "R/3," or non-ABAP types such as "J2EE"). After an eCATT run, ST30 immediately triggers the collector to gather DSRs of all known components. The system then collects all DSRs of the test time interval, or only those DSRs caused by the context of the eCATT test run (depending on the setting of the indicator *W/out Transact. Context*).

☑ Code Inspector

This tool (transaction SCI), which is also integrated into the ABAP Workbench, is used to perform static checks on development objects, such as programs, function groups, classes, and database tables. ST30 uses integrated Code Inspector features to:

- Check dynamic SQL (that is, the system examines the resulting SQL trace)
- Automatically detect and check the static program code involved
- Include the following Code Inspector checks in automated performance tests: *Appropriate Indexes*, *Complete WHERE clauses*, and *Buffers and Caches*

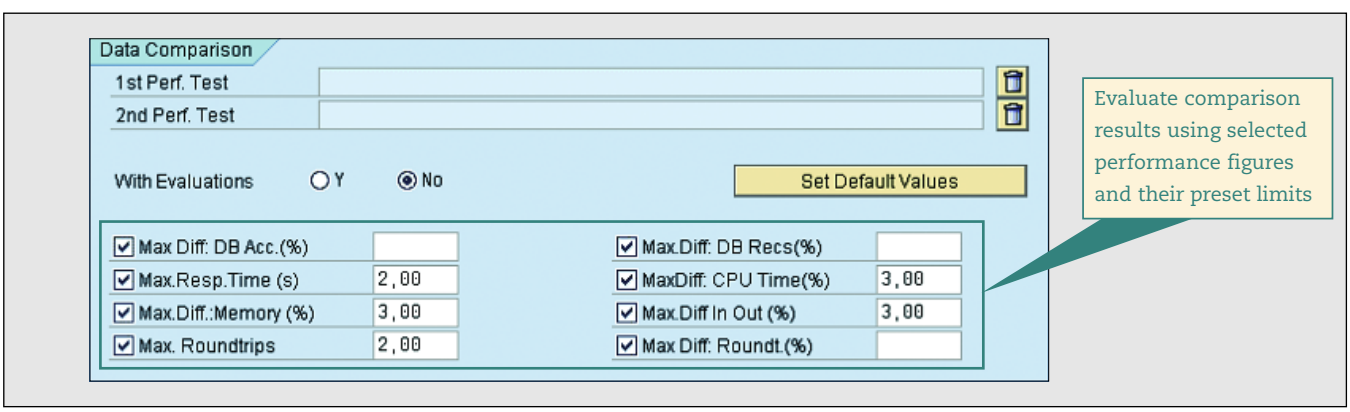


FIGURE 5 ▲ Compare performance data from different performance tests with automated data comparison

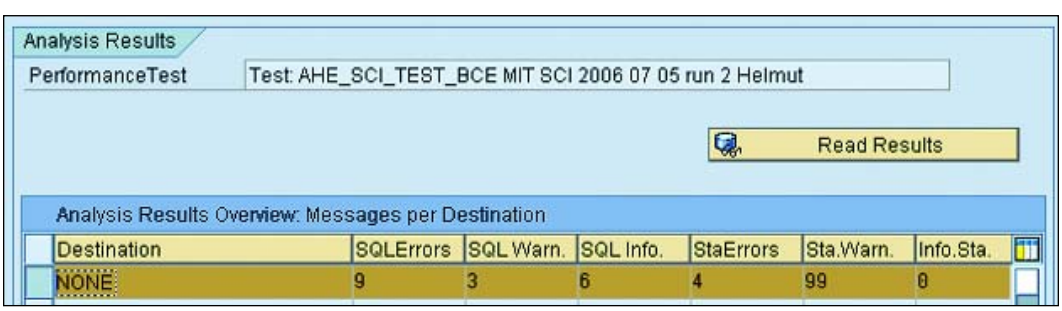


FIGURE 6 ◀ Analysis Results display the messages per destination

problem originated. Read the documentation provided for each performance problem category to find out more about the nature of the problem.

SAP's Performance Testing Outlook

SAP development is currently working on several new or enhanced ST30 features, which are expected to be available with an upcoming SAP NetWeaver release. These features include:

- An **automated validation** of performance test series available within ST30. The system will be able to determine whether test series meet the conditions in the checklist columns *Parallel processing enabled*, *Linear dependency memory*, and *Linear dependency time*.[7]

- A new pair of eCATT statements – **PERF** and **ENDPERF** – will be introduced within eCATT test scripts. You will be able to declare that only the eCATT commands between these two statements should be tested for performance rather than the whole script. Thus, it will be much easier to build repeatable scripts containing parts that are not supposed to be measured by a performance test – for initialization, for example.

- Transaction **SCEN_TEST**, a new test tool that integrates ST30 functionality. This tool, which SAP development is currently implementing, will enable the validation of performance test series for scalability tests and will be available with the next major release of SAP NetWeaver.

- Transaction **SE30** (ABAP trace) integrated as part of transaction ST30.

✔ **NOTE!**
Analysis data from the Code Inspector is only available if you set the Code Inspector-relevant indicators in the *eCATT Test* tab before running the performance test.

[7] For more information about running performance test series, see "Lower TCO and a Better End-User Experience Through Automated Performance Testing and Analysis" by Armin Hechler-Stark in the April-June 2006 issue of *SAP Insider* (www.SAPinsideronline.com).

Summary

Automated performance testing and faster, more cost-efficient systems go hand in hand. You may not know it, but you already have the tool you need to get started with automation; by setting up and using transaction ST30, you'll eliminate tedious, manual performance testing from your to-do list. ■

Additional Resources

- "Lower TCO and a Better End-User Experience Through Automated Performance Testing and Analysis," by Armin Hechler-Stark in the April-June 2006 issue of *SAP Insider* (**www.SAPinsideronline.com**)

- ST30 documentation in your system's SAP Library or in the SAP Help Portal (**http://help.sap.com**). As of SAP NetWeaver 2004s (SPS 09), follow this menu path for the Help Portal: *SAP NetWeaver Library* → *SAP NetWeaver by Key Capability* → *Solution Life Cycle Management by Key Capability* → *Testing* → *Test Workbench* → *Global Performance Analysis*