

BC System Services



HELP.BCCSTADM

Release 4.6C



Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.







HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

BC System Services	8
Tools for Monitoring the System	9
Overview of SAP Application Servers	10
Communication Table	13
Displaying and Controlling Work Processes	16
Controlling and Checking Processes (Menu Option Process)	19
Controlling the ABAP Program (Program menu option)	21
Editing Processes (Menu Option Edit)	22
Adjusting the Priority for Executing Work Processes (AS/400)	23
Displaying and Managing User Sessions	25
Displaying all the Users in Your System	27
Displaying Users on Another Application Server	28
Controlling User Sessions	29
Monitoring User Sessions	30
Displaying the User's Memory Usage	32
User Trace	34
Using Trace Facilities	36
Using System Traces	37
Configuring Tracing	39
Using Developer Traces	40
System log	44
Concepts of System Logging	45
Working with the System Log	46
Displaying a Log Report	47
Reading a Log Report	49
Using the Expert Mode	51
Configuring the System Log	53
Setting Central Logging Parameters	54
Understanding Central Logging Concepts	56
Specifying Log File Pathnames	58
Starting the System Log Processes	59
Terminating Processes	61
Troubleshooting Central Logging	63
Debugging Central Log Processes	65
Controlling Log Switching	66
Scheduling System Logging in the Background	67
Utilities	68
Consistency Check	70
Processing R/3 System Messages	71
Creating and Sending System Messages	73
Creating a System Message Under Program Control	74
The SAP Lock Concept (BC-CST-EQ)	76
Functions of the R/3 Lock Concept	77
The Owner Concept	81

The Lock Table	84
Lock Collisions	87
Cumulation of Locks.....	90
Questions and Answers Regarding Locks.....	92
Important Profile Parameters for the Lock Concept.....	96
Managing Lock Entries	98
Choosing and Displaying Lock Entries	100
Deleting Lock Entries	103
Testing Lock Management.....	105
Displaying Lock Statistics.....	106
Analyzing and Rectifying Problems	108
Updates in the R/3 System (BC-CST-UP).....	109
Functional Description of Updates.....	110
Update Request	112
Collective Runs	114
The Update Process	115
Bundling Updates.....	118
Synchronous and Asynchronous Updates.....	119
The Most Important Update Statuses	120
Error Handling and Data Consistency	122
Distributed Processing of Updates	123
V1 and V2 Update Modules.....	124
Update Dispatching with Load Balancing.....	125
Main System Profile Parameters for Updates.....	128
Additional System Profile Parameters	131
Reporting Update Problems.....	133
Automatic Update Stop in the event of Database Problems	134
Update Management	135
Selecting and Displaying Updates	137
Displaying the Update Header.....	139
Displaying Update Modules	141
Analyzing Canceled Updates	142
Testing Canceled Updates.....	143
Processing Updates Manually.....	144
Resetting Update Statuses	146
Deleting Update Records.....	147
Reorganizing Update Records.....	148
Configuring Update Groups	149
Displaying and Resetting Update Statistics	152
Displaying Servers	154
Update Program Administration (Transaction sm14).....	155
Documentation not available for Release 4.6C	157
Documentation not available for Release 4.6C	158
Documentation not available for Release 4.6C	159
Documentation not available for Release 4.6C	160
Documentation not available for Release 4.6C	161
Documentation not available for Release 4.6C.....	162
Documentation not available for Release 4.6C.....	163
Documentation not available for Release 4.6C.....	164

Documentation not available for Release 4.6C	165
Documentation not available for Release 4.6C.....	166
Documentation not available for Release 4.6C.....	167
Documentation not available for Release 4.6C.....	168
Setting Up Locals Executables on UNIX R/3 Instances	169
Structure of Executables after Standard Installation	170
Executables, Difference between Central and Local.....	171
Functions of the Automatic Adjustment.....	172
Installing Local Executables with Automatic Adjustment.....	175
Program sapcpe.....	177
Configuring sapcpe.....	179
Managing Batch Input Sessions	181
Process Overview: Batch Input	183
Batch Input: Concepts	186
Processing Sessions Automatically	188
Selecting Sessions and Reading the Session Display.....	189
Starting Sessions Explicitly: Run Modes	192
Analyzing Sessions.....	194
Correcting a Session	196
Deleting Sessions	200
Locking and Unlocking Sessions	201
Releasing and Restarting an Interrupted Session.....	202
Displaying Queue Management Information	204
Displaying Session Logs.....	205
Reorganizing the Batch Input Session Log File.....	206
Maintaining Tables	207
Overview of Tables.....	208
Client-Specificity: Maintaining Tables	210
Maintaining and Displaying Table Contents.....	211
Extended Table Maintenance	213
Overview	214
Concept	215
Authorizations	217
Generate Extended Table Maintenance Dialog	218
Call And Operation	220
Processing Mode.....	221
Create Sub-Work Areas of Records	222
Display Functions	226
Maintenance Functions	227
Transport.....	230
Security Audit Log	233
The Design of the Security Audit Log	235
Comparing the Security Audit Log and the System Log	238
Maintaining Static Profiles	240
Changing Filters Dynamically	242

Defining Filters	244
Displaying the Audit Analysis Report	246
Reading the Audit Analysis Report	248
Deleting Old Audit Files	250
Security Alerts in the CCMS Alert Monitor	251
Viewing Security Alerts	252
Reading Security Alerts Using BAPIs	253
The Audit Log Display Options	254
Example Filters	255

BC System Services

[Werkzeuge zur Systemüberwachung \[Page 9\]](#)

This documentation explains basic functions in R/3 Basis and gives an introduction to tools for monitoring, controlling and troubleshooting in the R/3 System.

The documentation is divided into the following sections:

Tools for Monitoring the System

[The R/3 Lock Concept \[Page 76\]](#)

[Updates in R/3 \[Page 109\]](#)

[Managing Batch Input Sessions \[Page 181\]](#)

[Utilities \[Page 68\]](#)

[Standard Table Maintenance \[Page 207\]](#)

[Extended table Maintenance \[Page 213\]](#)

Trace Functions

System logs

[Setting Up Local Executables on UNIX R/3 Instances \[Page 169\]](#)

[Security Audit Log \[Page 233\]](#)

Tools for Monitoring the System

Use

In the R/3 System, you find a set of tools for displaying detailed information on user sessions, work processes, and on the servers in your R/3 System.

If you want to work with these tools, choose in the R/3 initial screen *Tools → Administration*, or enter Transaction *s002*. The initial screen for system administration appears.

Features

The following functions are available:

[Overview of SAP application servers \[Page 10\]](#)

[Displaying and controlling work processes \[Page 16\]](#)

[Displaying and managing user sessions \[Page 25\]](#)

[Trace Functions \[Page 36\]](#)

[System Log \[Page 44\]](#)

Overview of SAP Application Servers

Overview of SAP Application Servers

Use

You can display a list of application servers that have registered themselves with the SAP message server by executing Transaction **SM51**, or by choosing *Monitor* → *System monitoring* → *Servers*. Only these application servers are active in an R/3 System.

You can also display the status of and manage users and work processes in any of the application servers that make up an R/3 System.

Integration

The function for managing users and work processes corresponds to the user- and process-tools. For more information, see [Displaying and Controlling Work Processes \[Page 16\]](#) and [Displaying and Managing User Sessions \[Page 25\]](#).

Features

The application server display includes the following information:

- **Server Name:** name of the individual application servers.
The name usually consists of the host name, the R/3 System name, and the R/3 System number.
- **Host:** host name of the individual servers
- **Type:** SAP [work process \[Ext.\]](#) type(s) for which an application server has been configured

The following functions are available as pushbuttons or in the menu:



You can use your cursor to select an application server by clicking on the relevant line (except for the *Refresh* function). The function that is executed then refers to this server.

Function	Meaning
Refresh	Refreshes the display
Processes (or double-click)	Displaying and controlling work processes [Page 16]
User	Displaying and managing user sessions [Page 25]
System log	System logs [Page 44]
OS collector	Statistics about the host platform, on which an SAP instance is running (for more information, see Operating System Monitor [Ext.] in the CCMS guide).
Remote logon	Log on by any server in the system; you can see the server where you are logged on in the status bar in the second entry from the left.

Overview of SAP Application Servers

Release notes	Display detailed release information for an application server (R/3 Kernel, database, operating system)
---------------	---

There are additional functions that you can only use via the menu:

Edit → Sort

This sorts the list alphabetically by the column on which the cursor is placed. The default is sorting by server name (first column).

Goto → SNC status

You can display information on the SNC connections of the server here.

The table contains the following fields:

Field	Meaning	Profile parameter
Server name		rdisp/myname
SNC active	Green light: yes / no light: no	snc/enable
Appl. PName		snc/enable snc/identity_as
Unsecure GUI logon allowed	Lock open: unsecure connections are accepted Lock closed: unsecure connections are not accepted	snc/accept_insecure_gui
Unsecure RFC logon allowed		snc/accept_insecure_rfc
Unsecure internal RFC logon allowed		snc/accept_insecure_r3int_rfc
Unsecure CPIC logon allowed		snc/accept_insecure_snc
Unsecure external programs allowed		snc/permit_insecure_start
GSSAPI library	Path to GSS library	snc/gssapi_lib

The profile parameters are explained in the RZ11 of the parameter documentation.

The following documents are also relevant for SNC:

- *SNC User's Guide*, Technical Document.
SAPNet: <http://sapnet.sap.com/systemmanagement> -> Security -> Secure Network Communications
- *SAP Complementary Software Program*
SAPNet: <http://www.sap.com/csp>
SNC, see: <http://www.sap.com/products/compsoft/scenarios/bc/bcsnc.htm>

Goto → SAP directories

The SAP directories of the server on which the cursor is currently positioned are displayed. The environment variables are displayed on the left and the path to the physical directory on the right. To display the contents of a directory, double-click that directory, or click the *Display* button. You can also display the individual files here. For more detailed information, see the CCMS guide.

Overview of SAP Application Servers

Goto → Communication table

The [communication table \[Page 13\]](#) for the relevant application server is displayed.

All the CPI-C connections for the selected server are displayed (see [BC - SAP Communication: CPI-C Programming \[Ext.\]](#) and [BC - SAP Communication: Configuration \[Ext.\]](#).)

The connections are displayed where the local dispatcher of the client is located and those connections where the server is located. This means, the connections to the server that were made externally.

Goto → Queue Information

Information about the Request queue for the relevant server is displayed, which means the number of configured work processes for the different work process types and statistical information about the number of requests.



Request type	Req. waiting	Max. Req. Wait	Max. Req	Req. written	Req. read
NOWP	0	19	2,000	602,859	602,859
DIA	0	23	2,000	242,075	242,075
UPD	0	2	2,000	95	95
ENQ	0	0	2,000	0	0
BTC	0	1	2,000	5,825	5,825
SPO	0	0	2,000	0	0
UP2	0	0	2,000	0	0

Goto → Environment

This displays the environment variables for the selected server.

Host name buffer → Reset → Appl. server OR

Host name buffer → Reset → Entire system

This option lets you make the changes made in the OS hosts and services tables in a running R/3 System take effect by making the desired changes at the operating system level and then, instead of restarting, resetting the host name buffer for the server or the entire system. (See also R/3 Note 25917.)

Activities

To display the initial screen for System Administration from the initial R/3 screen, execute Transaction **so02**, or choose *Tools → Administration*.

From the initial System Administration screen, execute Transaction **sm51**, or choose *Monitor → System monitoring → Servers*.

You can display this screen from any R/3 screen by executing Transaction **/nsm51**.

Communication Table

Definition

The communication table contains all of the CPI-C connections for the server on which the cursor was positioned (see [BC - SAP Communication: CPI-C Programming \[Ext.\]](#) and [BC - SAP Communication: Configuration \[Ext.\]](#).)

The connections are displayed where the local dispatcher of the client is located and those connections where the server is located. This means, the connections to the server that were made externally.

Use

The communication table helps you keep track of the CPI-C connections when monitoring the system or analyzing problems.

Structure

Destination	Conv-Id	User	Type	Status	Client	Req	Wp	Time
GTADIR_SERVER	61004034	KUNZ	CLIENT	ALLOCATED			1	09:16:44
	60788713	RFC_CORR_REQ	SERVER	ALLOCATED	R/3	CMRCV	0	09:18:22
EU_SCRP_WN32	60230342	KUNZ	CLIENT	ALLOCATED			1	09:05:18
	02602553	HINZ	SERVER	ALLOCATED	R/3		0	09:51:07
	88959741	SCHMITT	SERVER	ALLOCATED	R/3	CMSEND(SAP)	1	17:51:27
	86810585	MAIER	SERVER	ALLOCATED	R/3	CMSEND(SAP)	0	12:41:09

The entries in the table have the following meaning:

Field	Meaning
-------	---------

Communication Table

Des tin ati on	<p>Communication partner</p> <p>If the computer where you are currently logged on is the server of a CPI-C connection (see Type column), the destination column will be empty. If the local host is the CPI-C client, the remote destination, which is the server for this connection, is displayed in this field. You can display and maintain RFC destinations using Transaction SM59 or by clicking display and maintain Destination (see RFC Programming in ABAP [Ext.]).</p>	
Con v- Id	Conversation ID	
Use r	User	
Typ e	Client that requested a service and therefore the connection, or server that fulfills the client's request	
Sta tus	Current connection status; the following entries are possible:	
	INITIALIZED	Connection is being made
	ALLOCATED	Connection is made
	DEAL NOW	Connection is being terminated
	DEALLOCATED	Connection has been terminated
	CLEAR SNC	When using SNC: Encrypted data is still being supplied before the connection can be terminated.

Communication Table

Client	If Type = SERVER, information about the client. Is the client also an R/3 System or an external program?
Req	Last call or request (on the CPI-C level)
Wp	Last work process, used for processing
Time	Time of the last request that dealt with the connection

Displaying and Controlling Work Processes

Use

Work processes do the majority of the processing of the R/3 System. They execute dialog steps in user transactions, updates, lock administration, etc.

You can also find the term [Work Process \[Ext.\]](#) in the glossary.

You can display the current status of the work processes on the application server where you are logged on by choosing *Monitor* → *System monitoring* → *Process overview* or Transaction **sm50**. You must refresh the display to get updated information. The information on this screen is described in the following section.

The *Process overview* is intended primarily for information-gathering. For example, you can monitor processes to determine if the number of work processes in your system is adequate, to gather information for trouble-shooting, or for tuning.

Integration

By choosing *Monitor* – *System monitoring* – *Servers*, this displays the [overview of the SAP application servers \[Page 10\]](#). Here, you can further display the work process overview for a particular server in the R/3 System by clicking on the desired server name.

If system load is low, you may notice while using the *Process overview* that your requests appear to execute in only a single work process. The dispatcher is trying to use one work process for as many dialog steps for one user as possible. This avoids having to reload the roll area for the user.

Features

The *Process Overview* displays the following information:

- *No.:* The internal ID number of a process. Used to identify messages that pertain to a work process in the system log.
- *Ty.:* The type of work process:
 - DIA:* Work process for executing dialog steps in user transactions
 - UPD:* Update process for making U1 (time-critical) database changes
 - UP2:* Update process for executing U2 (not time-critical) database changes
 - ENQ:* For locking or releasing SAP lock objects
 - BTC:* For processing background jobs
 - SPO:* For spool formatting processes
- *PID:* Process ID of the work process (on the operating system)
- *status:* The current state of the work process. Possible statuses are:
 - Running* (executing a request).
 - Waiting* (idle and waiting for work)
 - Hold* (held for a single user)

Displaying and Controlling Work Processes

HOLD is not an abnormal state, but a work process that is in HOLD is restricted to serving a single user.

If too many processes are in *hold* status, then system performance suffers. You can use the *Reason* column to identify *holds* that can be released.

Ended (aborted with *Restart* set to *No*)

- **Cause:** If a work process is in HOLD status, displays the reason for the HOLD. Typical causes are: debugging; CPIC activity; enqueue (lock) activity; update activity; GUI (wait for response from the SAPGUI front-end program, for example, when waiting for a remote function call (RFC) to the front end).

You may also see PRIV (PRIVATE use) as a reason for holding a work process. PRIV indicates that a work process is reserved for a single user for memory management use. The work process has exceeded the limit of the SAP memory that is used by other processes. The process is held as long as the current user requires local memory. For more information, see [Private Memory \[Ext.\]](#) in the documentation on SAP Memory Management.

If a certain percentage of work processes are in PRIV status, the PRIV transactions are automatically ended when the user is not active in the transaction for a set period of time. These thresholds can be set in the R/3 system profile.

- **Start:** Indicates whether the process should be automatically restarted in the event of an abnormal termination. You can change the restart status of a process by choosing *Process* → *Restart after error* → *Yes/No*. Normally, leave *Restart* set to *Yes*.

If a work process aborts during its startup, the system automatically sets *Restart* to *No*. This measure protects against endless attempts to restart when the database system is not available or some other fundamental problem is affecting the system. After correcting the problem, you can change *Restart* to *Yes* so that the system starts the work processes.

- **Err:** Indicates how many times a work process has aborted
- **Sem:** Indicates the number of the semaphore for which a work process is waiting.
Normally, this field should remain empty. If one or more semaphore numbers frequently appears, evaluate the performance of your system using the Performance Monitor.
- **CPU:** Cumulative CPU time since the start of a work process. The time units are seconds and hundredths of seconds.



Calculating CPU time is expensive. Therefore, you must request this information using the *CPU* function.

- **Time:** Indicates the elapsed time used by a work process for the dialog step that it is currently processing
- **Program:** ABAP program or report that is currently being executed
- **Client:** The client of the session that is currently executing.
- **User:** User whose request is currently being processed

Displaying and Controlling Work Processes

- *Action*: Action that is being executed by the current program. The actions displayed are recorded by the R/3 Performance Monitor. The performance monitor must be active (R/3 profile parameter stat/level = 1 (default)) for actions or *Table* accesses to be displayed.
- *Table*: If the database is being accessed, this column shows the name of the table being accessed.

The menu offers the following functions:

- [Controlling and checking processes \(menu option Process\) \[Page 19\]](#)
- [Controlling the ABAP program \(menu option Program\) \[Page 21\]](#)
- *End session*: Deletes the user session being processed by the chosen work process (Corresponds to the garbage can icon on the menu bar)
- [Editing processes \(menu option Edit\) \[Page 22\]](#)
- *Goto*: You can display detailed information about the process, or user information on the relevant user, or go back to the previous screen (corresponds to the green arrow).



To manage users, use the *User Overview* ([Display and manage user sessions \[Page 25\]](#)). In this function, you cannot be sure that a user session you wish to cancel or delete is still executing in the work process that you select. You could unintentionally affect another user's session.

Controlling and Checking Processes (Menu Option Process)

Use

You can use this function to:

- Cancel a process (with or without core)
- Activate/deactivate the restart option after an error
- Execute various functions for the process trace

Prerequisites

You are already in the *Process Overview* (Transaction **SM50**) and have selected a work process using your cursor.

Features

The following functions are available:

Process → Cancel with core

You cancel a work process; a core file is created, which you can view using Transaction **ST11**.

Process → Cancel without core

You cancel a work process, and a core file is not written.

Process → Restart after error → Yes / No

You can choose whether a process should be restarted automatically after an error has occurred. The default is Yes.

Process → Trace → Active components

You can specify which R/3 components write trace information to the trace file and the trace level for logging (0: No trace, 1: Only trace errors, 2: Complete trace, 3: Complete trace with buffers). Only the *Taskhandler* component is set as the active default, and the trace level is 1.

Process → Trace → Display file

This displays the trace file for a selected work process.

Process → Trace → Reset files

This deletes the contents of the trace file that is continuously written until the system stops.

Process → Trace → Display settings

There are 2 options under this option, *Loading components* and *Display components*. These options are useful for very large trace files where all or many of the components were active. *Loading components* lets you choose which components are loaded in the file; *Display components* lets you further limit the display. The *Display components* are part of the loading components. By default, the loading components are displayed.

Controlling and Checking Processes (Menu Option Process)***Process → Trace → Dispatcher***

Here you can set the trace level for the `dev_disp` dispatcher trace file or display the trace file .

For more information on trace files, see [Developer Traces \[Page 40\]](#).

Controlling the ABAP Program (*Program* menu option)

Use

You can use this function to:

- End an ABAP program that is running.
- Debug an ABAP program that is running.

Prerequisites

You are already in the *Process Overview* and have chosen using your cursor a work process where an ABAP program is being executed (name is displayed in the column *Report*).

Features

The following functions are available:

***Program* → Debugging**

You can debug the ABAP program. For more information, see [Debugger \[Ext.\]](#).

***Program* → Cancel**

You can cancel the ABAP program; a short dump is processed.

Editing Processes (Menu Option Edit)

Editing Processes (Menu Option Edit)

Use

You can use this function to:

- Update the display
- Display the CPU time
- Cancel or exit the transaction

Integration

You can perform all of these functions by using the menu options, or by using the buttons on the menu bar: Refresh button, CPU button, the yellow arrow, and the red “X”

Adjusting the Priority for Executing Work Processes (AS/400)

Adjusting the Priority for Executing Work Processes (AS/400)

You have 2 options to adjust the priority for executing work processes:

- By adjusting the priority in the job description (JOBID)
- By using the new, relative priority classes

The priority specified in JOBID must be between 15 and 80. If this is not the case, a warning is given and the priority in JOBID is set to 20 (for priority <15) or to 80 (for priority >80).

You can improve the adjustment of the job priority by using the new job priority classes. There are 4 relative priority classes that are based on the priority defined in JOBID.

Priority class ID	Priority in relation to the value in JOBID	Meaning
HH	JOBID value -10	Highest priority
H	JOBID value -5	Highest priority
M	JOBID value -10	Medium priority
L	JOBID value +5	Low priority

Before you can use the new priority classes, you must activate the system value `QDYNPTYSCD` (Dynamic Priority Scheduler) by setting the value to 1. You require a system IPL (initial program load) to make the changes take effect.

You can then specify priority classes for the processes Update (UPD), Batch (BTC), and Spool (SPO). All the other processes have a fixed relative priority. The critical processes Message Server (MS), Dispatcher (DISP), Enqueue (ENQ) and Gateway (GW) are assigned priority class HH, for example. The user cannot change this. The following table displays which type of priority classes are permitted for which work processes. The work processes in **bold** can be changed by the user.

Priority class ID	Work Process
HH	MS, DISP, GW, ENQ
H	UPD
M	UDP , DIA, UPD2, BTC , SPO
L	BTC , SPO

Transaction **RZ11** lets you adjust the priority for the UPD, BTC, or SPO processes. The following parameters are relevant:

Parameter	Meaning
Parameter <code>rdisp/prio/upd</code>	Priority class for UPD
<code>rdisp/prio/btc</code>	Priority class for BTC
<code>rdisp/prio/spo</code>	Priority class for SPO

These parameters have *M* as the default value.

Adjusting the Priority for Executing Work Processes (AS/400)

You have activated the new priority classes. Your job description (JOBID) displays a priority of 20. Set UPD to priority class H, BTC to priority class M, and SPO to priority class L. The actual priorities for the various work processes are as follows:

Work Process	Priority of Execution
MS, DISP, GW, ENQ	10
UPD	15
BTC, DIA, UPD2	20
SPO	25

If **QDYNPTYSCD** = 0, all the processes have the default value for the priority of execution that is set in JOBID.

If **QDYNPTYSCD** = 1, new priority classes are always activated. If an incorrect class was specified for a changeable work process, its default value is assigned to this process and a warning is given.

Displaying and Managing User Sessions

Use

You can display all the users active in the system who are logged on to the application server by choosing *System monitoring* → *User overview*, or Transaction **sm04**. The server name is displayed in the status bar at the bottom of the screen.



Client :	User	Terminal	Transaction	Time	Sessions	Ty.
000	MUELLER	p23879		13.53.27	1	RFC
000	MUSTER	p25581	SM51	11.14.16	1	GUI
000	SCHMIDT	p21223	SE38	14.21.44	1	GUI
000	MAIER	p25556	SCEM	14.29.39	1	GUI
000	SCHMITT	p23838	SM04	13.12.10	1	GUI
000	GRAF	p25593		14.33.03	1	GUI
000	HANSEN	p31057	SEU_INT	10.25.36	1	GUI
000	FISCHER	p24129	SE38	14.33.42	2	GUI

*** 8 users logged on with 9 sessions ***

The following table explains the meaning of each column.

Client:	R/3 client
User	User logged on to server (R/3 user name)
Terminal	Terminal at which the user is working. (If it is a UNIX frontend, the terminal name corresponds to the display variable of the frontend process; if it is a Windows or OS/2 frontend, the terminal name corresponds to the host name on which the frontend was started.)
TCode	Last executed R/3 transaction (transaction code)
Time	Time at which the user last initiated a dialog step by entering data
Sessions	Number of external sessions (session [Ext.]) opened by the user (up to 6) You can display detailed information on a session by choosing the user with the <i>Sessions</i> function.
Ty.	Type of connection (GUI or RFC)

Integration

You can call the User Overview from the [SAP application server overview \[Page 10\]](#) by choosing *Goto* → *User*, or by executing Transaction **SM51**.

Displaying and Managing User Sessions

Features

The following functions are available as menu options. Important menu options are also available as buttons on the menu bar.

[Display all users in your system \[Page 27\]](#)

[Display users on another application server \[Page 28\]](#)

[Controlling user sessions \[Page 29\]](#)

[Monitoring User Sessions \[Page 30\]](#)

Activities

The User Overview appears:

- From any R/3 screen by using Transaction */nsm04*
- From the initial R/3 screen by choosing *Tools → Administration → Monitor → System monitoring → User overview*
- From the system administration screen by choosing *Monitor → System monitoring → User overview*.

Displaying all the Users in Your System

Use

When you start the *User overview*, the system displays to you the users who are logged on to the same application server as you are. You can however, display all the users logged on to the R/3 System.

Prerequisites

You are already in the *User List* screen (Transaction **SM04**).

Procedure

Choose *Goto* → *Terminals* to display a list of all users who are logged on in your R/3 System. The list includes all application servers in your system.



In this display, you can look at users, but you cannot access user sessions. To display information or access a user session, switch to the server on which the session is active. Here, you have the following options:

- Double-click on a sever name.
- Position the cursor on a desired server and choose *Goto* → *User* → *Servers*.

If you choose *Goto* → *User* → *Local*, the user overview of the server you are logged on to appears.

To display the users on another server from the user overview, see [Displaying Users on Another Application Server \[Page 28\]](#).

Displaying Users on Another Application Server

Displaying Users on Another Application Server

Use

You can display a user list from other servers in an R/3 System using this function.

Prerequisites

You are already in the *User List* screen (Transaction **SM04**).

Procedure

Choose *Goto* → *User* → *Servers*. In the dialog box that appears with the server list, choose the desired server and click *Choose*.



You switch over to the new server. You can see which server you are logged on to in the third entry in the status line on the bottom right. If you click the green arrow, or choose *Goto* → *Back*, this takes you back to your old server.

Controlling User Sessions

The *User overview* allows you to intervene in the user sessions that are executing in work processes. You can do the following:

- Delete a user session.
Deleting a user session terminates the session and the user transaction executing in it.
Deleting a user's last session terminates the user's current operation and logs him or her off the system.
- Switch a user session into debugging session.
The system starts the ABAP/4 debugging facility and locks the work process for the exclusive use of the user session that you have switched into debugging mode. You can then analyze the execution of the transaction.
- Terminate the program that is running in a user session.
Terminating a program stops it, causing an abnormal termination of the program.
- Take over a user's session.
If a problem occurs in a user's front-end server (PC or workstation), the user can sometimes lose access to his or her session in the R/3 System. The user's SAPGUI front-end program has stopped running, but the user's session is still active in the R/3 System.
If this problem occurs, a user can resume working in a lost session by "taking over" the session. When a user takes over a session, he or she resumes working in the session as though the front-end problem had never occurred.
A user can take over a session when he or she logs on to the R/3 System again. To do this, type in your user ID and password and then select *User → Take over session* rather than simply pressing ENTER. Or a user can take over a session in the User overview with *Process → Take over session*.
If the user was in the middle of a dialog step (logical unit of work (LUW)) when the SAPGUI problem occurred, then this dialog step is lost when the user takes over a session. The session is resumed in the state it was in when the user's context was last rolled out from a work process. Any actions taken in the current dialog step cannot be recovered.
In general, little or no information is lost when taking over a session, and it is easy for a user to resume work. This is because a new dialog step is started whenever a user presses ENTER or selects a function key or menu. Often, a dialog step is created to carry out a trivial action, such as executing a *Page down* in a list. Taking over a session when the last action was *Page down* means only that the cursor is positioned as it was before the page down.

You can intervene only in your own user sessions unless you have administrator authorization (*System Administration Functions* authorization object).

Monitoring User Sessions

Monitoring User Sessions

Use

When you monitor a system, you can use this function in addition to the *User Overview* to analyze the system's work load via the users. You can:

- Filter and sort the list
- Update the display
- Display user information
- Display the memory requirements for the users
- Activate/deactivate and display the user trace

Prerequisites

You are already in the *User List* screen (Transaction **SM04**).

Features

The following table gives you an overview of the monitoring functions and the menu paths (and buttons).

Function	Menu Path	Button Available?	Use
Sort	<i>Edit → Sort</i> (F7)	Yes	Position the cursor on the column that should be sorted and choose <i>Sort</i> . Note that uppercase letters come <i>before</i> lowercase letters.
Filter	<i>Edit → Filter</i>	No	A dialog box appears where you can enter the filter conditions for the client, the user name, and the transaction code. Click the <i>Filter</i> icon and a filtered user list appears.
Refresh	<i>Edit → Refresh</i>	Yes	Refreshes the display
User info	<i>Goto → User info</i>	Yes	Dialog box appears displaying user information; this is maintained in the User Maintenance (Transaction SU01) (see Creating Accounts [Ext.] in the CCMS documentation).
Sessions	<i>Goto → Sessions</i> (Shift F6)	Yes	Displays (external) sessions that a user has opened; transaction codes may also be displayed. The system administrator can delete individual sessions (click <i>End session</i> in the dialog box <i>Overview of Sessions</i>).
Display user's memory usage [Page 32]	<i>Goto → Memory</i>	No	Displays user memory usage; the used memory of all actions (divided into different types) for a user is added up.

Monitoring User Sessions

User trace [Page 34]	<i>Edit → Trace</i>	No	For troubleshooting
--	---------------------	----	---------------------

Displaying the User's Memory Usage

Displaying the User's Memory Usage

Prerequisites

You are already in the *User List* screen (SM04; *User list* screen).

Procedure

You can display the usage of the various [R/3 memory types \[Ext.\]](#) by choosing *Goto* → *Memory*.

Client:	User	Transaction	Roll	Page	Mem (Total)	Mem (Priv.)
000	MUELLER	SE38	458.752	57.344	5.225.243	0
000	MUSTER	SESSION_MANA	262.144	0	1.161.577	0
000	SCHMIDT	SE01	458.752	81.920	4.551.960	0
000	MAIER	SE11	1.048.576	172.032	13.572.756	0
000	SCHMITT	SESSION_MANA	262.144	0	1.161.577	0
000	GRAF	OS_APPLICATION	851.968	147.456	18.664.742	0
000	HANSEN	OS_APPLICATION	851.968	139.264	16.085.115	0
000	FISCHER	SE61	458.752	65.536	3.463.431	0

*** 8 users logged on with 9 sessions ***

The following table explains the meaning of each column.

Client:	R/3 client
User	User logged on to server (R/3 user name)
Transaction	Executed R/3 transaction (transaction code)
Roll	The R/3 roll area [Ext.] holds the user environment information needed by work processes when executing dialog steps.
Page	SAP Paging is only used for a limited number of ABAP commands.
Mem (Total)	Specifies how much of the SAP extended memory [Ext.] was requested by this user.
Mem (Priv.)	Private memory [Ext.] (heap memory) requested by the user.

The memory usage for all the work processes currently occupied by this user is displayed.

You can use transaction SM50 ([Displaying and Controlling Work Processes \[Page 16\]](#)) to monitor the memory usage of *individual* work processes.

The order in which the various memory classes are assigned to the dialog work process is described under [Allocating Memory for User Contexts \[Ext.\]](#).

The display also lists the number (key) assigned to the user. Roll and page blocks are identified internally by a key that comprises the key number of the user who holds the block and the number of the user's mode and the number of the user's internal mode.

Displaying the User's Memory Usage

For further information on R/3 memory management, please refer to the documentation on [BC - Memory Management \[Ext.\]](#).

User Trace

User Trace

Use

The user trace is used to look for errors that occur with certain users and not with others. Trace information, located in the [developer trace \[Page 40\]](#) for each work process, is written to the user trace from the user view, if this option is activated.



Activate the user trace only as long as you need it to reproduce error situations, and then deactivate it. This prevents system performance from being affected, and unnecessarily large trace files from being produced.

Integration

For more information, see [Trace Functions \[Page 36\]](#).

Prerequisites

You are already in the *User List* screen (Transaction **SM04**).

Features

You can do the following with the user trace:

- Activate it (*Edit* → *Trace* → *Trace on*)
- Deactivate it (*Edit* → *Trace* → *Trace off*)
- Display it (*Edit* → *Trace* → *Display trace*)

If you select the function *Display*, a dialog box appears where you can enter information to filter the trace display. You can also set the *loading components* (R/3 Kernel components that should write trace information) and the *display components* (the parts of the loading components whose trace information is also displayed).

Trace Display

In the trace display (screen *User Trace for <USER> on <server>*), there are additional menu functions and buttons available.

User Trace

```

User Trace for HAUGT on ds0024 AIO 32
Trace Edit Goto System Help

Display components
Step hv user      HAUGT      Session 1 . Step 1
Step bv user      HAUGT      Session 1 , Step 2
Step bv user      HAUGT      Session 1 . Step 3
M step 3 with ra id 5983 started in wp 2 Thu Sep 17 11:05:24 1998
M ra id 5983 for T77 U4282 M0 I0 (Mstat 18. len 125) (from dispatcher) Tname: p24129
M ThRnInCheck: o.k.
M Adresse  Offset  Data from TM/WP
M -----
M x0180ea60  000000  00001100 00000001 c4000000 121f9d02 |.....|
M x0180ea70  000016  a7ca3158 611088a1 e0445db6 51c12022 |..1Xa....D1.Q. "|
M -----
M new request: THFCTERM
M TskhLoop: set task type to dia
M PfRecCreate: create record (0)
M RollIn: install saved spa pointer 1409177e0
M ThCheckEmState: call EmContextAttach (em hdl=64)
X EmContextAttach (64)
I Sem2Lock: LockObiPtr = 39d958. Yield = 0
I Sem2Unlock: LockObiPtr = 39d958
M RollIn: no roll in necessary
M RollIn: roll in abap all
M RollIn: call rtscb call back (back after)
M uncompress data from 133 to 196 bytes
M Thdynpen00: call dynpen00 (2) ...
Y
Y ===== DYNP entrv 2:
Y RollEnvir 04282 00 0000000048 00 HAUGT
Y receive data from SAPGUI
D DiagInputAppStUser.ST USER GUI VERSION=45B
D DiagBracket : CHLN occurred
D DiagMainDvno : Dvno WRows 34
D DiagMainDynpro : Dynp WCols 118
D DiagMainDvno : List WRows 46
D DiagMainDvno : List WCols 119
D DiagBracket : CONTAINER_RESET occurred

```

You can:

- Expand and collapse the individual steps (double-click on the step or the left green or red '+'/'-' button on the menu bar)
- Expand or collapse all the steps at once (right '+' or '-' button on the menu bar)
- Output the trace information without a format (*Trace* → *Unformatted* or *Shift + F2*)
- Set the load and display components (*Trace* → *Component selection*)
- Reset the trace file (delete) (*Edit* → *Reset trace*)
- Run a function trace (*Edit* → *Function trace*)
- Go to the next C-stack (*Edit* → *Next C stack*)
- Go to other system functions (*Goto*)

Using Trace Facilities

Using Trace Facilities

[BC - ABAP Workbench: Werkzeuge \[Ext.\]](#)

[BC - ABAP Workbench: Werkzeuge \[Ext.\]](#)

With the trace facility, you can trace four kinds of operations in your R/3 System:

- SQL database accesses
For more information, please see the SQL trace documentation in the *ABAP/4 Development Workbench Tools* guide. You'll find this guide in the Basis System Administration documentation section on the documentation CD-ROM.
- ABAP/4 programs
For more information, please see the ABAP/4 runtime trace documentation in the *ABAP/4 Development Workbench Tools* guide. You'll find this guide in the Basis System Administration documentation section on the documentation CD-ROM.
- internal operations in the R/3 System
- traces generated by R/3 processes (developer traces).
These are the trace files that are written by the individual host system processes that make up an R/3 System. These trace files contain highly technical information for use in the event of problems in your System.

To reach the trace functions, select *Tools* → *Administration* → *Monitoring Traces*. Note that the R/3 ABAP/4 Development Workbench provides access to these traces and to additional trace and debugging tools.

[Using System Traces \[Page 37\]](#)

Configuring Tracing

[Using Developer Traces \[Page 40\]](#)

Using System Traces

[Systemlog \[Page 44\]](#)

[Entwickler-Trace \[Page 40\]](#)

Use the Internal trace function to trace internal R/3 System activity in the application server in which you are logged on. You can start tracing by selecting one of the entries in the Set menu or from the trace options screen, described in "Setting Internal Trace Options" below.

Generally, you should start tracing by setting the trace options that you need in the trace options screen. If you start from the Set menu on the main screen, then your trace includes all active users, which can affect System performance.

To display a trace, select *Analyze* → *Analyze standard*. You can obtain more detailed information and explanations on any entry by selecting the entry.

Information in trace entries includes the following:

- *Last sync*: The time specification in the left-hand column is the time elapsed since the last sync point was set. The System sets sync points at such events as the start of a program or task switch.
- *T.(Type)*: The type of trace entry. The types correspond to the options in the trace options screen.
- *specific part*: The trace message text.

You may see formatted trace output, in which the message text starts with a set of activity codes. You may also see unformatted trace output, in which no codes appear.

The codes in formatted trace output are as follows:

- V: Start of a discrete function with trace output extending over one or more additional lines.
- ^: End of a discrete function. The message also includes the elapsed time from the start to the end of the function.
- ->FB: Call to a function module.
- <-FB: Return from a function module with the time spent in the module.
- X: A discrete function, generating only a single trace record.
- M->M: Start of a module written in C. (Not all C modules write these entries.)
- M<-M: End of a module written in C. (Not all C modules write these entries.)
- ABAP: Entries generated from a function written in ABAP/4.
- C: Entries generated from a module written in C. (Not all C modules write these entries.)

You can get brief explanations of the codes by selecting an entry.

All internal traces are written in a circular file. When the file is full, then the System writes trace entries from the beginning of the file again.

If more than one trace is included in the file, the display includes a message indicating where the new trace began.

Using System Traces

Setting Internal Trace Options

The System sets internal trace options automatically whenever you start a trace from either the SQL trace or Trace internal functions. However, you can also start these trace functions by setting trace options yourself and then starting the trace from the trace options display. In particular, you use the options to limit a trace to a particular user or program.

If you start an SQL trace from this menu, however, you will be able to display the trace only with the internals-trace report, not with the special SQL trace report.

Start trace options by selecting *Traces* → *Internal trace Set* → *Change switches*.

The topics which follow explain how to use several of the options for controlling internal traces.

Starting and Stopping a Trace

You start a trace when

- you set options manually and press the ENTER key
- you set options with any of the functions provided in the *Set* menu.

With the Trace function, you can mark all trace options, including *Write to disk*.

With the All types function or the Total trace cross-off option, you can mark all trace options except write to file. You should always write traces to file. Otherwise, they are not displayable.

You can stop all tracing with *Set* → *Stop trace*. Active programs do not write any more trace entries and flush their trace buffers to the trace file. You can turn off tracing entirely with *Options* → *No trace*.

When you stop a system trace, you should first select Stop trace and then after a short pause, No trace. This sequence allows the system trace to write any remaining buffered entries to file and to shut down in an orderly fashion.

Writing a Trace to File

To display a trace, you must have the trace written to file. SQL traces are automatically written to file. To have an internal trace written to file, mark the Write to disk option in the options display.

With the Write to disk option set, the system buffers trace entries and writes them to file each time that the buffer fills. You can then display the trace with the system trace Analyze functions.

Unbuffered Write to File

Should you wish to do so, you can have entries written directly to disk, without buffering. To do so, mark the Unbuffered write to disk option. Use this option with caution: your System's performance will be noticeably slowed by the extra file activity.

Flushing Buffers

You can force the contents of all trace buffers to be written to disk by marking a cross on the Write now option. Each traced program flushes its buffer to disk immediately, or, if inactive, as soon as it becomes active again.

Each R/3 process in an application server has its own trace buffer.

Configuring Tracing

With system profile parameters, you can set the maximum size of the SQL and internals trace file and specify its path name. All trace parameter names begin with the string `rstr/`. For default values, please see the system profile maintenance function.

The trace parameters that you can set are as follows:

- `rstr/max_diskspace`: Specify the maximum amount of disk space that can be allocated to the trace file.

When the file is filled, the System starts writing trace entries from the start of the file again.

- `rstr/file`: Specify the absolute path name of the trace file. By default, the central shared log directory of an R/3 System. The trace file name is `TRACE<R/3 System number>`.

The System maintains additional trace parameters which you should not change.

Using Developer Traces

Using Developer Traces

Developer traces contain technical information for use in the event of problems with your System. Using the entries in the developer traces requires a sophisticated knowledge of the host systems in which your R/3 System is running and of the R/3 System itself.

The traces can be useful in diagnosing host system and R/3-internal problems that are affecting your R/3 System. For example, the disp+work process reports an inability to fork, or create, work processes in the developer trace.

Developer traces are written in files in the `work` directory of the R/3 application server that generated the trace.

Turning Developer Traces On and Off

You can turn developer traces on and off and set the trace level dynamically from within R/3 or with system profile parameters or command-line arguments.

Activating / Deactivating Developer Traces from within R/3:

1. Choose *Administration* → *System management* → *Monitoring* → *System monitoring* → *Process overview* (alternative: transaction SM50).
2. Choose the work process in which you wish to increase the trace level. To trace all work processes of a server, use the system profile method shown below.
3. Choose *Process* → *Trace* → *Active components*.
4. The system presents a dialog screen that shows the current status of the developer trace.

Turn developer tracing on and off for different server components by marking the appropriate boxes.

Set the amount of detail by entering a number in the *Level* field. Possible trace levels are as follows:

- **0**: No trace.
- **1**: Write error messages in the trace file.
- **2**: Full trace. The trace entries that are actually written can vary with the R/3 program that is being traced.
- **3**: Additionally, trace data blocks.

Activating / Deactivating Developer Traces from the System Profile:

You can also set trace options instance-wide with the `rdisp/TRACE=<n>` option. The trace values are the same as those in the list above.

Setting the trace parameter in the instance profile of a particular server activates developer traces only in that server.

Setting the parameter in the default profile would activate traces in all servers (excepting those whose instance profiles specify a different parameter value).

Activating / Deactivating Developer Traces from the Command Line:

You can turn off developer tracing or set various levels of tracing with options in the command line of any R/3 program. Typically, you must add these options to the start commands for R/3

Using Developer Traces

processes in the start-up profile of an instance of your R/3 System. For information on editing start profiles, see the profile maintenance documentation in the *Computing Center Management System Guide*.

The command-line options are as follows:

- **TRACE=0:** No trace.
- **TRACE=1:** Write error messages in the trace file.
- **TRACE=2:** Full trace. The trace entries that are actually written can vary with the R/3 program that is being traced.
- **TRACE=3:** Additionally, trace data blocks.

Displaying Developer Traces

You can display developer traces from within R/3 and from the operating system level.

From R/3:

1. Choose *Administration* → *System management* → *Monitoring* → *System monitoring* → *Servers* (alternative: transaction SM51).
2. Select the server whose developer traces you want to display. Then choose *Goto* → *Traces*.
3. The system presents the list of developer traces available at the server. (The names of the files are listed below.)

Choose *Attributes* to display summary information on the trace, including the user who created it and the time stamp.

Choose *Display* to display its contents.

You can also display work process traces from the *Process overview* function (transaction SM50) in the same menu path.

From the operating system:

1. Log on to the correct host system.

You need to access the `work` directory of the R/3 application server whose traces you wish to see. This directory is always local, located on the host on which an R/3 application server is running. You'll therefore need to log on to the host system of the R/3 application server.

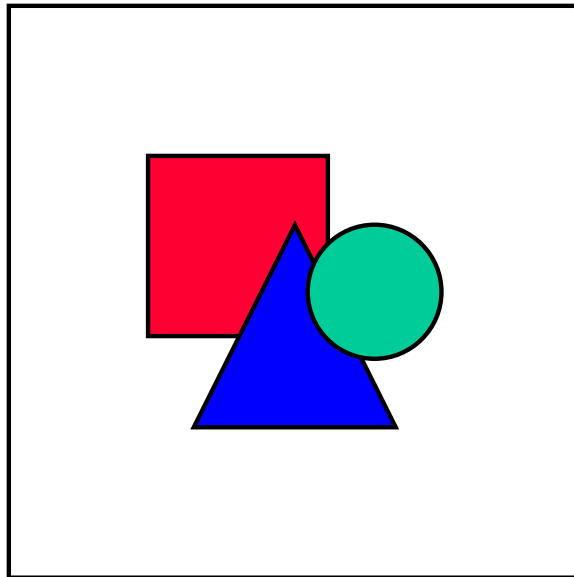
2. Change to the `work` directory of the R/3 application server.

Typical menu path (UNIX; other systems analogous): `/usr/sap/<SYSTEM ID>/<INSTANCE>/work`
`/usr/sap/CLP/D01/work`

3. Display the list of files in the directory. (The names of the files are listed below.) Use a host system command like `more` to display the files.

Always display trace files from within the R/3 System if you can. The files are presented in R/3 in an easier-to-read format than they are at the operating system level.

Using Developer Traces



Core files, profiles, and start traces: Should one of the work processes of an application server terminate abnormally with core dump, you will also find the core files in the work directory.

Further, you can display the start-up and instance profiles used by an R/3 application server along with the developer traces. Server start-ups are recorded in the `sapstart<n>.trc` files.

File Names of Developer Traces

The names of the trace files are as follows:

Developer Trace File Names

Component	File Name
Dispatcher	dev_disp
Message server	dev_ms
Work process:	
Task handler	dev_w<n> n is the range from 0 to one less the number of work processes.
Dynp (screen processor)	dev_dy<n>
Roll	dev_ro<n>
Paging	dev_pg<n>
DB interface	dev_db<n>
ABAP processor	dev_ab<n>
Enqueue (lock)	dev_eq<n>
Logging	dev_lg<n>

Using Developer Traces

Spool	dev_w<n> In the trace file, spool messages have an ID beginning with S.
Message server	dev_ms<n>
SAPGUI (presentation)	dev_st<logon name>
APPC-server (CPIC gateway)	dev_appc
RFC (Remote Function Call) facility	dev_rfc, dev_rfc<n> dev_rfc traces RFC calls to external functions (written in C or Visual Basic). dev_rfc<n> traces RFC calls that are executed in R/3 work processes. <n> is the number of the work process in the server (as shown above). A work process uses the same log file across RFC calls.
Gateway	dev_rd
R3trans and tp transport programs	dev_tp
Monitoring infrastructure (test mode only)	dev_moni In normal operation, you will not see this file. It is used only by test tools of the monitoring infrastructure. It therefore appears only if the test tools must be activated during a support session.

Error and System-Log Entries in Trace Files

In the files, lines that contain error information begin with ***** ERROR =>**. Lines for which system log entries are written start with ***** LOG <message ID>**.

An "error line" contains the name of the calling function, the operation that failed, the error number from the host system (if a system call is involved), and the name of and line in the C module that failed.

System log

System log

You can use the log to pinpoint and correct errors occurring in your system and its environment.

To work with system logs, choose `Tools` → `Administration` from the R/3 initial screen. When the System Administration initial screen appears, choose *Monitoring* → *System log*. *Alternatively, you can choose Test* → *System log from the ABAP/4 Development Workbench initial screen.*

Working with the System Log

[Displaying a Log \[Page 47\]](#)

[Reading a Log Report \[Page 49\]](#)

[Expert Mode \[Page 51\]](#)

Configuring and Running the System Log

[Specifying Log File Path Names \[Page 58\]](#)

[Understanding Central Logging Concepts \[Page 56\]](#)

[Setting Central Logging Parameters \[Page 54\]](#)

[Starting the System Log Processes \[Page 59\]](#)

[Terminating Processes \[Page 61\]](#)

[Troubleshooting Central Logging \[Page 63\]](#)

[Debugging Central Log Processes \[Page 65\]](#)

[Controlling Log Switching \[Page 66\]](#)

Concepts of System Logging

R/3 servers record events and problems in system logs. *If your R/3 System runs on UNIX hosts, then* there are two types of system logging: local and central. Each R/3 application server has a local log that contains messages generated by this server. You can also enable central logging. With central logging, each application server also copies its local log entries to a central log.

On Windows NT and AS/400 hosts, only local logs (one per application server) are kept. These logs are not collected into a central log.

How the System Writes Log Files

Each R/3 application server maintains a single local log. System messages are logged in a single ring buffer on the server. When this log file reaches the maximum allowable size, the system starts overwriting the file from the beginning.

If you like, you can configure your server so that messages are issued to a [central log file \[Page 56\]](#) as well as to a local log. (Central logging is not available on AS/400 and Windows NT host systems.) A central log file is maintained on a selected application server. Individual R/3 application servers (or instances) send their local messages to this single system. This system collects the messages from each instance and writes them to the central log.

A central log is written in two files: an active file and an old file. The active file contains the current log. When the active file reaches the maximum length specified in the system profile, the system performs a log switch. The system deletes the old log file, uses the active file as old file, and creates a new active log file.



The system does not notify you when it replaces an old log.

Accuracy of Log Files

A local log is always up to date. This is not always true for a central log. There can be brief delays between when the system records a message in a local log and when the same message is written in the central log. Delays can occur because the send processes on local systems sleep between periods of activity. The period of inactivity varies with the message volume on a particular system.

Central log files can take longer to build when a message is sent from a system that normally has little log activity. Longer delays and lost messages on local hosts can result from major network disruptions or failure of the collection process on the machine containing the central log.

Working with the System Log

[Displaying Logs \[Page 47\]](#)

[Reading a Log Report \[Page 49\]](#)

[Expert Mode \[Page 51\]](#)

Displaying a Log Report

To display system logs, choose *Tools* → *Administration* from the R/3 initial screen. The system displays the System Administration initial screen. Choose *Monitor* → *System log*. The System Log initial screen appears.

The *System log entries imported* field shows how many log entries the system has read from the log file. When you first enter the System Log initial screen, this field is set to 0. Choose *Reread system log* to read the log. The system then reads in entries from the log file so that you can display them. By default, the system reads the log for the last one to two hours.

If you have read a log, the toolbar contains *Redisplay only* and *Read in Syslog* buttons along with the *Refresh Syslog*:

Choose *Redisplay only* to view the last system log you displayed. If you like, you can use the *Selection* options along with this button to refine the old system log data without re-reading the log.

If you want to retain the data from this last read, but want to reread the system log with different selection criteria, use the *Read in SysLog* option. This option lets you mix the information from an original log access with a second log access into one list.

Setting Selection Criteria

You can tailor the information you want to pull from the log using the selection criteria.

Reading Logs on Remote Systems

If you like, you can read the system logs on other application servers in your R/3 system. Choose *System log* → *Choose* to select a log. You can display the following logs

Log	Displays
<i>Local system log</i>	Messages issued by the application server you are currently logged onto. This is the default.
<i>Remote system log</i>	The system log from an application server other than the one you are logged onto. To display a particular remote system, use the <i>Instance name</i> option.
<i>All remote system logs</i>	The logs from the system you are logged onto as well as from all the other application servers that make up that R/3 System.
<i>Central system log</i>	<p>The central log [Page 56]. To use this option, you must have previously configured each application server to forward the messages from its local log to the central log.</p> <p>The information in the central system log may differ from the data in the individual system logs. See Concepts of System Logging [Page 45] for information about the log file accuracy.</p>

Environment Utilities

The *Environment* menu provides the following functions:

<i>Display SAPPARAM</i>	Displays the settings of system log parameters in the system profile.
-------------------------	---

Setting the Log Entry Presentation

<i>Show authorizations</i>	Shows the authorizations required for the system log.
<i>Clocks</i>	Displays the system clocks that were used for message time stamps. Checking system clocks is useful if you need to clarify time discrepancies between messages from different application servers. The display shows whether the system clocks of your R/3 systems are set to the same time.
<i>Process status</i>	Displays the current state of the system log's send processes.

Setting the Log Entry Presentation

Use the selection screen to determine what information is displayed for the individual log entries. To do this, choose *Goto* → *Layout*. In the [Expert Mode \[Page 51\]](#) , you can also display the *Development class* and the *Date*.

Reading a Log Report

To display a current system log, choose `Reread system log` from the System Log initial screen. The system displays a table with the system log entries. The information available with this table depends on your selection criteria. If you used the default selection criteria, your system log appears similar to the following:

SysLog: lokale Auswertung auf hs0311

2

Zeit	Typ	Nr	Man	Benutzer	Tcod	MNr	Text
17:19:07	BTC	5	000	DDIC		Q01	Betriebssystem Aufruf gethostbyname gesc
17:19:07	BTC	5	000	DDIC		Q09	Hostname p33918 nicht bekannt
17:19:07	BTC	5	000	DDIC		Q01	Betriebssystem Aufruf gethostbyname gesc
17:19:07	BTC	5	000	DDIC		Q09	Hostname p33918 nicht bekannt
17:19:09	BTC	5	000	DDIC		Q01	Betriebssystem Aufruf gethostbyname gesc
17:19:09	BTC	5	000	DDIC		Q09	Hostname p22107 nicht bekannt
17:19:09	BTC	5	000	DDIC		Q01	Betriebssystem Aufruf gethostbyname gesc
17:19:09	BTC	5	000	DDIC		Q09	Hostname p22107 nicht bekannt

The title page of the system log includes a summary of your selection criteria. The last page of the log is a table of contents for the log.

If you select the *with statistics* option in the `Format` text box, the system includes a statistical analysis. This analysis includes a report of message frequency by client, transaction, report, and user.



An empty *User* field in a log report means that the system issued the message during startup or shutdown of the R/3 System. There are some other system activities that are not assigned to an individual user.

Display Details

You can display detailed information on a specific entry by selecting the entry and choosing *Edit* → *Details*.

Use *Edit* → *Analyze errors* to display a detailed error analysis in the detailed view if short dumps or runtime errors have occurred.

Sorting the System Log Display

You can use the *System Log* → *Sort* function to sort a log report by various criteria.

Messages and the Log Display

Choose *Edit* → *Ignore message* if you want the system to ignore a message type. The next time the system processes the system log data for output, the system ignores messages of this type. When you use the *Ignore message* function, the system automatically enters the relevant

Reading a Log Report

message ID into the message ID exclusion/inclusion screen. This screen is only visible when you are working in [Expert Mode \[Page 51\]](#) .

Using the Expert Mode

The System Log utility has two modes: normal and expert. The expert mode is intended primarily for use by SAP should it become necessary to analyze a problem in your system. To enter expert mode, choose *Edit* → *Expert mode*.

In expert mode, you have a number of additional selection options you can use. You can also exclude or include specific system log messages.

Setting Expert Selection Options

Choose the *Attributes* button to display the following fields:

Attribute	Description
Program	Restricts the display to messages from one program.
Problem classes	Limits the report to a specific message type. You can enter one or more of the following message classes: K system kernel messages from the Basis system S status messages T transaction messages W warning messages X other types of messages
From file/position To file/position	Defines the segments of the log file that are read. If you have already read the system log once, you can determine the file and position of a particular entry by double-clicking on the entry. An entry's detail page contains the entry's file and position in the <i>Technical Details</i> section.
Message format (Type)	Selects messages by system component format. To do this, enter the code letter for a component's SysLog type. To list component formats, use the <i>Possible Entries</i> button.
Terminal	Selects messages from single-place computers, such as PCs, enter the network hostname of the PC. See the F1 help for additional formats for search specifications.
Development class	Restricts the display to messages issued from modules in a development class. The TDEV table lists the available development classes.
With internal SysLog entries	Displays messages from the system log's send and collection processes. With these entries, you can check that the send processes in application servers are successfully communicating with the central collection process. See Understanding Central Logging Concepts [Page 56] for more information on the send and collection processes.

You can use the *Message IDs* function to exclude specific message types from a report or include only specific message types in a report.

Configuring the System Log

[Specifying Log File Path Names \[Page 58\]](#)

[Understanding Central Logging Concepts \[Page 56\]](#)

[Setting Central Logging Parameters \[Page 54\]](#)

[Starting the System Log Processes \[Page 59\]](#)

[Terminating Processes \[Page 61\]](#)

[Troubleshooting Central Logging \[Page 63\]](#)

[Debugging Central Log Processes \[Page 65\]](#)

[Controlling Log Switching \[Page 66\]](#)

[Scheduling System Logging in the Background \[Page 67\]](#)

Setting Central Logging Parameters

Setting Central Logging Parameters

On each application server central logging is inactive by default. Before you can enable central logging, you must ensure that the logging parameters in the central system and in each instance are identical. The system parameters that control central logging all begin with the **rslg** prefix.

To set the **rslg** parameters, you can edit the profile on each application server, or you can edit the DEFAULT.PFL file on the central application server. The recommended method is to edit the DEFAULT.PFL.

Editing the DEFAULT.PFL

To set the central logging parameters using the DEFAULT.PFL profile, do the following:

1. Display the **rslg** parameter values on the central system.

To do this, choose *Environment* → *Display SAPPARAM* from the System Log initial screen.

2. Check the values of the following parameters:

rslg/collect_daemon/host

rslg/collect_daemon/listen_port

rslg/collect_daemon/talk_port

rslg/send_daemon/listen_port

rslg/send_daemon/talk_port

These parameters must be set and identical on the central system and all the instances.

See *Central Logging Parameters* below for information on acceptable values for each parameter. Depending on your installation, many of the **rslg** parameters may already be set.


3. Edit the DEFAULT.PFL and ensure that all the parameters are correct.
4. Save your changes.

The system sets the **rslg** parameters on the central system and all the instances.




After you have ensured that the system profiles are set appropriately, you must start the send and collection processes. See [Starting the System Log Processes \[Page 59\]](#) for more detailed information

Central Logging Parameters

The **rslg** parameters have the following definitions:

rslg/collect_daemon/host	Specifies the node name where the central log will reside.
rslg/collect_daemon/listen_port	<p>Specifies the listen port for the collection process. By default, this parameter is set to:</p> <p>12< SAPSYSTEM number></p> <p> You must ensure that the listen port is identical for all instances in your R/3 system.</p>

Setting Central Logging Parameters

rslg/collect_daemon/talk_port	Specifies the talk port for the collection process. By default, this parameter is set to: 13< SAPSYSTEM number>  You must ensure that the talk port is identical for all instances in your R/3 system.
rslg/send_daemon/listen_port	Specifies the listen port for the send process. By default, this parameter is set to: 14< SAPSYSTEM number>  You must ensure that the listen port is identical for all instances in your R/3 system.
rslg/send_daemon/talk_port	Specifies the talk port for the send process. By default, this parameter is set to: 15<SAPSYSTEM number>  You must ensure that the talk port is identical for all instances in your R/3 system.
rslg/send_daemon_autostart	Leave the value of this parameter set to 0.

The send process also uses the following system profile parameters. Do not change the default values of these parameters:

rslg/collect_daemon/exe_file	Specifies the executable file for the central collection process.
rslg/send_daemon/exe_file	Specifies the executable file for the send process.
rslg/send_daemon/pid_file	Identifies the file that the send process initializes when it is started.
rslg/send_daemon/status_file	Identifies the internal status file maintained by the send process.

Understanding Central Logging Concepts

Understanding Central Logging Concepts



This section does not apply to R/3 Systems running on IBM AS/400 and Microsoft Windows NT hosts. Central logging is not available on these platforms. See below for details on central logging with Windows NT systems.

If you like, you can configure your server to write to a central log as well as to a local log. Central logging requires 2 or more R/3 instance or application servers. A single system is the central system. This system collects the log data from the other instance or application servers. For central logging to work, you must do the following:

- Set the logging parameters in each system profile.
- Start a collection process on the central application server.
- Start a send process on the central system and on each instance.

Central System Log Processes

To maintain a central system log, two processes are required: a send process and a collection process. Each instance uses the send process. From each instance, the send process copies local messages to the central log.

The send process is activated automatically at intervals. Each time a send is activated it forwards all the log messages that were posted since its last activation. After forwarding the local messages, the send process sleeps to save system resources.

The intervals between activations of the send process can be less than one minute or more than 30 minutes. The intervals vary automatically according to the level of log activity on an application server. If many entries were written in the log since the last activation of a process, then the next period of deactivation is correspondingly short.

A central instance uses both a send process and a collection process. The collection process receives messages transferred by the send processes on each instance. The collection process is activated in response to communication requests from send processes. Between communication requests, the collection process sleeps.

Process Status Files

Both the send and the collect system processes maintain a file with the extension **pid**. The **pid** file contains the operating system ID (process number) of the process. If a process's **pid** file contains -1, the process is not allowed to run. If a process's **pid** file is absent, the process was either never started or was terminated.

In addition to a **pid** file, a send process also maintains a file with a **sta** extension. The **sta** file contains the file descriptor of the file that the send is processing. The **sta** file also contains the byte offset up to which the process has processed in the file.



The byte offset is not updated with every system log message processed.

Central Logging with Windows NT

No central system log is kept on Windows NT systems. The processes necessary to maintain a central log are not required. To avoid system warnings regarding the central log, set the R/3 system parameter **rslg/collect_daemon/host** on your NT system to NONE.

Use the *System log* → *Choose* → *All remote system logs* function to read the data from all instances in your R/3 System. If an alert occurs in the Alert monitor, evaluate the affected instance using the *Remote SysLog* function.

Specifying Log File Pathnames

Specifying Log File Pathnames

You can specify the paths and file names for local and central system log files with the following system profile parameters:

Parameter	Definition
rslg/central/file	Specifies the active central log filename. The default filename is: SLOGJ
rslg/central/old_file	Specifies the old central log filename. When a log switch occurs, the active log file is copied onto the old log file. The default filename is: SLOGJO
rslg/local/file	Specifies the local log filename. The default filename is: SLOG<SAPSYSTEM number>

If necessary, you can edit these path names in the profile parameter maintenance. To do this, choose *Tools* → *CCMS* → *Configuration* → *Profile Maintenance*.

About the R/3 System Number

To specify log file path names, you can use the R/3 system number directly or you can specify the symbols \$\$ in the profile. A system's number is identified by the SAPSYSTEM parameter. This parameter is defined in a system's start-up and system profiles. The SAPSYSTEM appears in R/3 instance names.



hs1044_C11_17

C11 is the SAPSYSTEM. The host name is hs1044 and it is defined by the SAPDBHOST parameter. 17 is the R/3 System number. Another C11 application server in hs1044 must have a different R/3 System number.

Normally, all application servers of an R/3 System have the same R/3 number. Servers must have different numbers only if two servers that belong to the same R/3 system are to run in the same host. In this case, the number is used to differentiate between the servers.

Starting the System Log Processes

Once you have set the [Central Logging Parameters \[Page 54\]](#), you must start the R/3 system processes that send messages from local logs to the central log. On the central system, you start a collection process and a send process. On each instance, you must start a send process.



This section does not apply to R/3 Systems running on IBM AS/400 and Microsoft Windows NT hosts. Central logging is not offered on these platforms, and the processes described below are not needed.

Start a Collection Process on the Central Instance

You should start a collection process only on the instance where the central log is to be maintained. To start the collect process, add the following commands to the start profile of the instance:

```
#-----  
# start rslgcoll  
#-----  
_CO                =co.sap<SID>_DVEBMGS00  
# create a link name for rslgcoll  
Execute_05         =local ln -s -f $(DIR_EXECUTABLE)/rslgcoll  
    $(_CO)  
# start the program, specifying the instance profile of the server  
Start_Program_05   =local $(_CO) -F  
    pf=$(DIR_PROFILE)/<SID>_DVEBMGS00
```

Starting the Send Processes

Start a send process in each instance of your system, including the central system. The send process forwards messages from each instance to the central log. To start a send processes, add the following lines to the start profile(s) of each instance:

Starting the System Log Processes

```
#-----  
# start rslgsend  
#-----  
_SE                =se.sap<SID>_DVEBMGS00  
# create a link name for rslgsend  
Execute_06         =local ln -s -f $(DIR_EXECUTABLE)/rslgsend  
    $(_SE)  
# start the program, specifying the instance profile of the server  
Start_Program_06   =local $(_SE) -F  
    pf=$(DIR_PROFILE)/<SID>_DVEBMGS00
```



You can display the full syntax and options of RSLGCOLL and RSLGSEND by doing the following:

1. Change to the R/3 executables directory on a host system in which an R/3 application server is installed. Typical example under UNIX: /usr/sap/<SYSTEMNAME>/SYS/exe/run.
2. Enter the program name (rslgsend, rslgcoll) and help or ?:

```
rslgsend help
```

The program displays its syntax.

Terminating Processes



This topic does not apply to R/3 Systems running on IBM AS/400 or Microsoft Windows NT hosts. The log processes described in this topic aren't required on these platforms.

You should explicitly terminate the collect and send system log processes as part of any shutdown or reboot of the host operating system. If you start processes with the start-up profile, the processes are shut down automatically by a system shutdown. The log system processes are not affected by an abnormal termination of the R/3 System. Abnormal termination is termination of the work and dispatcher processes.

Should you need to shut down a log process with an operating system command, you can have the process perform a normal or an expedited orderly termination. An example of such an operating system command is the UNIX **kill** command.

Using a terminate option starts a normal termination. The affected process finishes any processing that is underway. The process then terminates rather than sleeps. In this case, a send process, for example, attempts to finish processing all new messages in a log before terminating.

Using a terminate option a second time starts an expedited termination. An active process completes its current unit of work and then terminates. A send process, for example, finishes sending the current message before terminating.



An inactive process is immediately terminated by a terminate command. The process will not restart automatically until the you reboot the system or manually restart the process.

Effects of Operating System Shutdowns

If you shutdown a system without terminating the collect process or if the system crashes, the current central log may become corrupted. Corruption occurs if the collect process is writing a message to the log file when the shutdown occurs.

If a log becomes corrupted by an incomplete message or other problem, you must manually correct the problem by repairing or deleting the log file. The log system processes respond with error messages to a corrupted file.

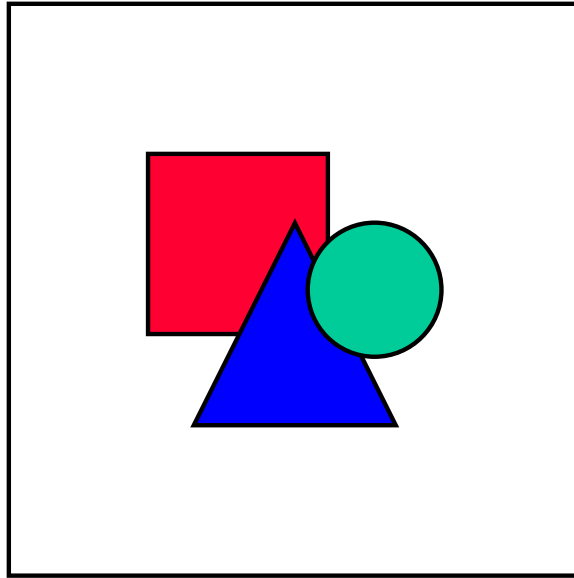
Locking and Unlocking Processes

Locking a process prevents it from being started. Send processes, in particular, do not start automatically if you lock them. You should lock a process when you do not want system log messages forwarded or if the entire R/3 System is running in a single instance. Use the following procedure to lock processes:

1. Stop the process that you wish to lock.
2. Enter the value **-1** in the process's **pid** file.

Terminating Processes

Troubleshooting Central Logging



This topic does not apply to R/3 Systems running on IBM AS/400 or Microsoft Windows NT hosts. The log processes described in this topic aren't required on these platforms.

If you attempt to display the central log file and receive the error *Syslog file not found*, you may have problems with your central log configuration. To verify if you have configured your central log system correctly, display the central log file from each server in your R/3 system. The central system log should contain the message:

Central system log is collected on <hostname>.

where <hostname> is the host you specified in the **rslg/collect_daemon/host** parameter.

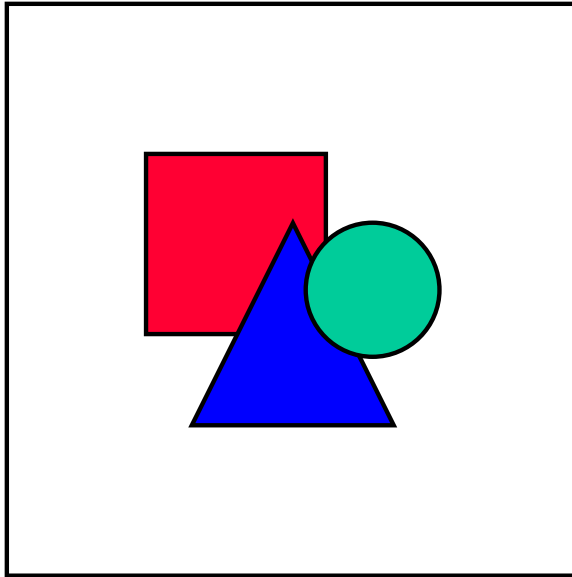
To verify your central log configuration, check the configuration on the central log host and each server in your R/3 system. Ensure that the following parameters are set correctly:

rslg/collect_daemon/host

rslg/collect_daemon/listen_port

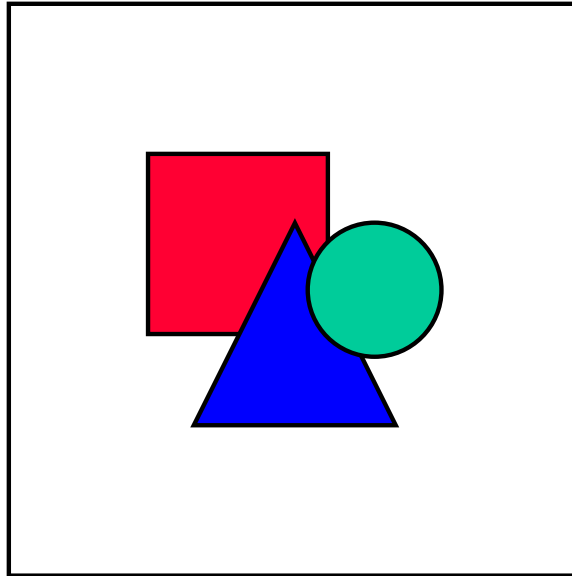
rslg/collect_daemon/talk_port

The SAPSYSTEM number in these parameters must be set to the central database server. If one or more instances have different SAPSYSTEM number values, then those systems are not automatically included in the central log. You can also ensure these values are identical, by [setting the parameters \[Page 54\]](#) in the default system profile, DEFAULT.PFL.

Troubleshooting Central Logging

If your R/3 system has only one instance, you have no need for a central log. In this case, set the parameter **rsig/collect_daemon/host** to NONE. Otherwise, when you attempt to display a central log, the system warns you that it can not find the central log.

Debugging Central Log Processes



This topic does not apply to R/3 Systems running on IBM AS/400 or Microsoft Windows NT hosts. The log processes described in this topic aren't required on these platforms.

R/3 provides developers and system programmers with debugging facilities for system log processes. The debugging facilities consist of starting a process in trace mode. When in trace mode processes, exhaustively log their activities to trace logs. These traces logs are separate from the system log and unique to the send and collection processes.

You can start the send and collect processes in trace mode using the following command line options:

-v2	Redirect through stderr to a special log.
-vt	Write as free-format c-User entries in the R/3 trace file. This option is valid only if R/3 Trace is activated.
-v	The same as specifying both -v2 and -vt .

Deleting Logs

If you use debug mode, trace output is not written in the system log. Instead, trace output is written into a separate trace log. The R/3 system does not restrict the length of these trace logs. Ensure that you delete these trace logs when you finish debugging. If you do not delete the trace log, you can encounter shortages in disk space with trace turned on.

Controlling Log Switching



This topic does not apply to R/3 Systems running on IBM AS/400 or Microsoft Windows NT hosts. The log processes described in this topic aren't required on these platforms.

Log switching occurs in central logging when the system replaces the old log with the active log. In local logs, log switching occurs when the system begins overwriting the log file. Central log switching is regulated by the **rslg/max_diskspace/central** parameter. Local log switching is regulated by the **rslg/max_diskspace/local** parameter.

The **rslg/max_diskspace/local** and **rslg/max_diskspace/central** parameters specify the length of log files. If these parameters value are large, switching occurs less frequently than if the value were smaller. Additionally, a large log size can help prevent lost messages in the event that sudden heavy message activity causes rapid switching of the central log.

To control central log switching, set the **rslg/max_diskspace/central** parameter to the maximum combined length of the current and old central logs. You must set this parameter in the system where the central log resides. The system switches the logs when the active log file size reaches approximately half the value specified in the **rslg/max_diskspace/central** parameter.



The **rslg/max_diskspace/central** and **rslg/max_diskspace/local** parameters do not affect the trace logs used by the log system processes to record their own error messages. These logs are not subject to automatic deletion; you must delete them yourself if you selected them using special options.

For more information on how the system writes log files, see [Concepts of System Logging \[Page 45\]](#).

Scheduling System Logging in the Background

Use

You also have the option to schedule system logging as a background job. There are two ABAP programs available for this:

- For creating a local system log: RSLG0000
- For creating a central system log (not on Windows NT and AS/400 platforms: RSLG0000)

Activities

1. Execute one of the above programs in the *ABAP Editor* (*Tools* → *ABAP Workbench*), set conditions if necessary and save your selection as a variant.
2. Schedule this variant as a background job using the transaction for [Background Processing \[Ext.\]](#) (*Tools* → *CCMS* → *Jobs* → *Definition*).

Result

The programs create either a printout of the system log or a spool request, depending on the settings you have selected.

Utilities

Utilities

Use

The R/3 System offers a variety of utilities for performing administrative tasks.

Integration

This chapter covers those system monitoring functions that have not already been described in a separate chapter.

Features

These utilities include the following:

- **System Messages**

With *Tools → Administration → System messages*, you can broadcast messages to all users who are logged on to an R/3 System.

For more information, see [Sending System Messages \[Page 71\]](#).

- **ABAP Short Dumps**

If you choose *Tools → Administration → Monitoring → Dump analysis*, you can display the dump summaries generated if an ABAP program terminates abnormally.

- **Status**

With *System → Status*, you can display the current status of the session in which you are active.

The status information includes the transaction code of a function. You can enter the transaction code in the command field as a short cut to the function. You can also use the transaction if you want to lock a function using Tcode administration.

- **Transaction Administration**

With *Tools → Administration → Administration → Tcode administration*, you can lock or unlock transactions. Transactions are the programs that are behind the menus of the R/3 System.

Locking a transaction prevents users from carrying out the function represented by the transaction.



For example, locking Transaction **SA38** (ABAP reporting) prevents users from running reports.

- **Consistency Check**

With *Tools → Administration → Administration → Installation check*, or using Transaction **sick**, you can check for inconsistencies in your system. The function is also called automatically whenever you start your system or start an application server. The function checks for the following:

- the release number in an R/3 kernel matches the release number stored in the database system
- the character set specified in the R/3 kernel matches the character set specified in the database system
- critical structure definitions that are defined in both the data dictionary and the R/3 kernel are identical. The structures that are checked include SYST, T100, TSTC, TDCT, TFDIR, and others.

- **Number Range Buffer**

If you choose *Tools → Administration → Monitoring → Number range buffer*, you can display status information on this buffer. The information is useful if you want to tune the buffer.

See also: [Number Range \[Ext.\]](#)

Consistency Check

Consistency Check

Use

The purpose of the consistency check is to determine inconsistencies in your system. The function is also called automatically whenever you start your system or start an application server.

Features

The installation check checks whether:

- the release number in an R/3 kernel matches the release number stored in the database system
- the character set specified in the R/3 kernel matches the character set specified in the database system
- critical structure definitions that are defined in both the data dictionary and the R/3 kernel are identical. The structures that are checked include SYST, T100, TSTC, TDCT, TFDIR, and others.

Activities

Choose *Tools* → *Administration* → *Administration* → *Installation check* or enter the transaction code `sick` (**S**AP **I**nitial **C**onsistency **C**heck**K**).

Processing R/3 System Messages

Use

You can use short messages to send important information to the users of an R/3 System. You can either display the messages to all users on all application servers, or only to the users who are logged on to a particular application server.

The system displays a message only once to each user during a logon session. Messages are displayed to users

- as each user logs on
- as soon as you post a message, for users who are already logged on.

A user must clear a system message from his or her screen before continuing to work.

Features

The system message facility offers the following functions:

- **Create message**

You can post as many messages as you wish. Messages can be more than one line long. This is described in detail in the section [Creating and Sending System Messages \[Page 73\]](#).

- **Display message**

Choose *Edit* → *Current messages* or *Archived messages* to display either the active messages or the inactive, archived messages (messages that are past their expiration date, but that are still available in the database).

- **Change message**

You can alter the text and expiration date of any message (see [Creating and Sending System Messages \[Page 73\]](#)).

- **Delete message**

You can delete a message/delete all messages. To delete a specific message, position the cursor on it and choose the 'Delete' button. You can also use *Edit* → *Delete* to do this, or use *Edit* → *Delete all* to delete all system messages.



Deleted messages are then removed from the database, which means they are also no longer kept with the archived messages.

- **Creating a message under program control**

Normally, you issue system messages interactively. However, it is also possible to do this from a program. See [Creating a System Message Under Program Control \[Page 74\]](#) for further details.

Processing R/3 System Messages**Activities**

Select *Tools* → *Administration* → *Administration* → *System messages* or enter transaction **sm02**.

Creating and Sending System Messages

Prerequisites

You are already in Transaction sm02 (the 'System messages' screen).

Procedure

Choose either the *Create* button or *Messages* → *Create* from the menu.

A popup will appear, in which you can make the following entries:

Systemnachrichten anlegen

Systemnachrichtentext

Servername

Mandant

verfällt am 31.03.1999 23:00:00

löschen am 30.04.1999 23:00:00

- Enter the text of the message at the top. The message text has a maximum length of 3 lines.
- You can then restrict the client and the server. If you enter nothing or a * here, the message will be sent to all users logged on to this system (regardless of client and server).
- The **expiration date** specifies as of what day the message will no longer be displayed to users. The message will however remain in the R/3 database and can be viewed until the **delete date** has passed. The default setting for the expiration date is the current day at 11:00 p.m., and the default setting for the delete date is one month later. This data can, however, be overwritten.

Result

You have created a system message. This will now be displayed to every user working in the system. Users receive all current system messages when they log on to the system.

This lasts until the expiration date of the message, or until it is deleted .

Creating a System Message Under Program Control

Creating a System Message Under Program Control

It can sometimes be useful to issue (create) system messages under program control rather than interactively. For example, you can issue warning messages from a background job before starting a database backup or other action that causes the R/3 System to pause.

Procedure

To issue a message under program control, you need to create a program that calls the function module shown below. When the program runs (for example, as a job step in a background job), the message that you specify is displayed immediately as a system message. The message expires and is no longer displayed at the time that you specify.

```
data:  emtext like TMSG-EMTEXT.
```

```
data:  exp_time like TMSG-TIMDEL.
```

```
* Messages have the same format as the dialog transaction --
* a maximum of 3 lines each with a maximum of 60 characters.  Only the
first line
```

```
* must contain a message.
```

```
emtext = "Bitte abmelden. Backup in 2 minutes.
```

```
* Enter the expiration time as shown. Use 12- or 24-hour format as *
set in the user master record under which the program will run.
```

```
* If only time is specified, today's date is assumed.
```

```
* Always specify a time.
```

```
exp_time = '230000'.
```

CALL FUNCTION 'SM02_ADD_MESSAGE'

EXPORTING

MESSAGE	= emtext	" 1. Zeile der Nachricht
MESSAGE2	=	" 3. Zeile
MESSAGE3	=	" 3. Zeile
SERVERNAME	=	" Server
EXPIRATION_DATE	= SY-DATUM	" Verfallsdatum mit
EXPIRATION_TIME	= '230000'	" Zeit
DELETE_DATE	= ' '	" Löschedatum mit
DELETE_TIME	= '230000'	" Zeit
CLIENT	= ' '	" Mandant

IMPORTING

MESSAGE_ID	=
------------	---

EXCEPTIONS

Creating a System Message Under Program Control

EMPTY_MESSAGE	= 1
SERVER_NOT_AVAILABLE	= 2
CLIENT_NOT_AVAILABLE	= 3
NOT_AUTHORIZED	= 4
OTHERS	= 5

Result

Each time your program runs, the message you specify is issued to all users.

The SAP Lock Concept (BC-CST-EQ)

The SAP Lock Concept (BC-CST-EQ)

Purpose

The SAP System is equipped with a special lock mechanism that synchronizes access to data on the database. The purpose of the lock mechanism is to prevent two transactions from changing the same data on the database simultaneously.

Implementation Considerations

Locks are defined generically as "lock objects" in the Data Dictionary. A *lock request* is a specific instance of a lock object and locks a certain database object, such as a correction or a table entry.

Lock entries are usually set and deleted automatically when user programs access a data object and release it again.

Integration

The SAP lock mechanism is closely related to the [update mechanism in R/3 \[Page 109\]](#). A description of handling lock objects is provided in the ABAP Dictionary Documentation under [Lock Objects \[Ext.\]](#).

The ABAP documentation explains the key elements of the lock concept with regard to programming ABAP transactions in the section entitled [The R/3 Lock Concept \[Ext.\]](#).

Features

You can use the [Lock Management \[Page 98\]](#) functions (transaction SM12) to check and delete lock entries if the SAP dispatcher, operating system, or network connection fails and the dispatcher is not able to delete these entries. In this case, invalid lock entries remain effective and block access to the locked data when the system is restarted.

For a better understanding of the R/3 lock concept, please refer to the section entitled [Functions of the SAP Lock Concept \[Page 77\]](#).

The most important profile parameters for the R/3 lock concept are described [here \[Page 96\]](#). You can use these parameters to tailor your system resources to your needs.

Functions of the R/3 Lock Concept

This section explains how the R/3 mechanism is implemented. It provides background information that will help you understand and apply the lock management concept. The specific options with regard to handling R/3 locks are described under [Managing Lock Entries \[Page 98\]](#).

R/3 Lock Logic

If an R/3 transaction is to make changes to database objects, the programmer of the transaction must lock the objects first to prevent concurrent access and then release them again.

To do so, he or she must *define* and *activate* a lock object in the Data Dictionary (see [Lock Objects \[Ext.\]](#) in the Data Dictionary documentation). Activating a lock object causes the system to generate two function modules: one for locking the object, and one for releasing it. This procedure is described in detail under [Lock Mechanism \[Ext.\]](#) in the ABAP Dictionary documentation.

Lock Server

In a distributed R/3 System, one lock server (also referred to as the *enqueue server*) manages the [lock table \[Page 84\]](#). This server runs on the central instance.

If a lock is to be set in an application running on an instance other than the central instance (for example, on a different host), the lock request is transferred via the [dispatcher \[Ext.\]](#) and message server to the dispatcher of the central instance, which then forwards it to the enqueue work process. This process then checks the lock table to determine whether the lock request collides with an existing lock (see also [Lock Collisions \[Page 87\]](#)). If this is the case, the request is rejected. Otherwise, the lock is set and an appropriate entry is made in the lock table.

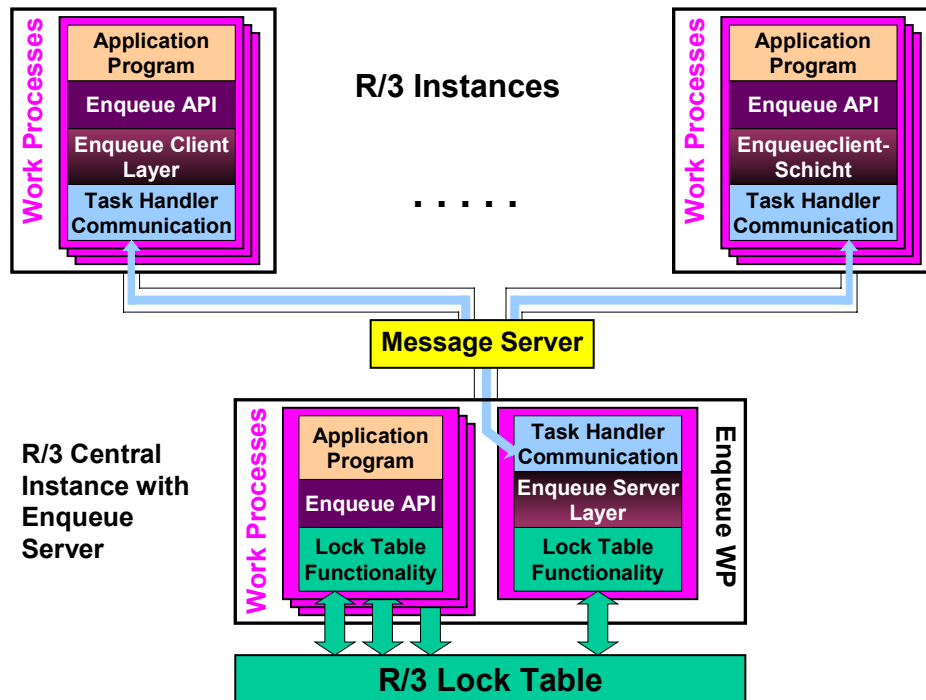


The work processes on the central instance have direct access to the lock table functionality. This means that they do not have to send their lock requests via the dispatchers and message servers.

The graphic below shows the communication path in a distributed R/3 System with one central instance and additional instances.

Functions of the R/3 Lock Concept

Enqueue communication



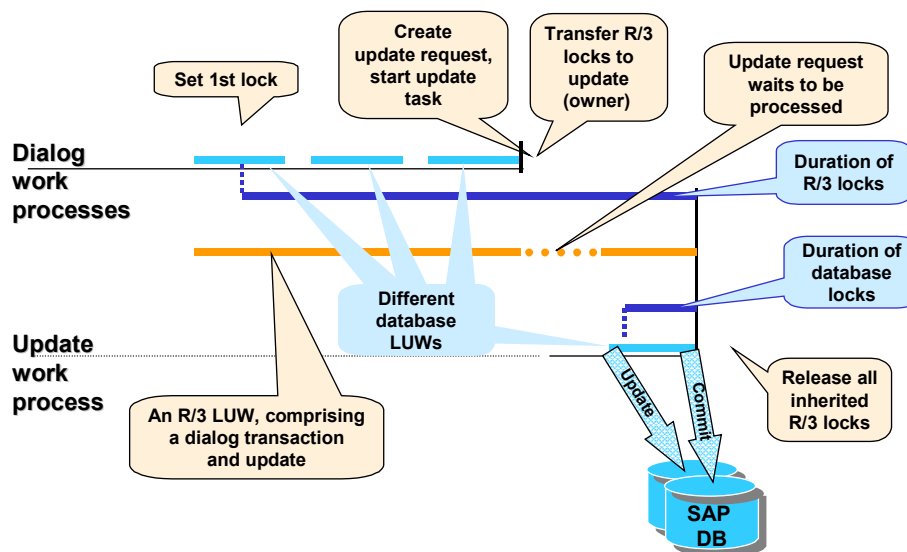
Locks and R/3 Update

During the course of the transaction, the lock is transferred to the [update in R/3 \[Page 109\]](#). This procedure is described in detail under [The Owner Concept \[Page 81\]](#) and [Function Modules for Lock Requests \[Ext.\]](#). Locks that have been transferred to the update are stored both in the lock table and in a backup file so that they are not lost if the enqueue server fails. The backup flag is then set in lock management.

SAP Locks and Database Locks

The following graphic shows the relationship between SAP locks and database locks. The objective here, naturally, is to minimize the duration of a database lock. In addition, unlike database locks, an SAP lock can exist across several database LUW. The difference between SAP LUW and database LUW is described under [Functions of the Update Task \[Page 110\]](#).

R/3 Transaction Concept: Minimize Database Lock Time



The top (solid) blue line represents an R/3 dialog transaction that comprises three screens (input windows); each screen corresponds to a database LUW (see [Functions of the Update Task \[Page 110\]](#)). Once the user has made an input, the database LUW (light blue line) ends.

Processing is then continued by a dialog work process. After the second user input, processing is completed and the dialog section of the SAP LUW is terminated.



The transaction does not have to be processed by the same dialog work process. With each screen, the dispatcher searches for a free work process to handle the processing.

In this example, an R/3 lock was requested in the first screen of the transaction - this lock remains in place (top dark blue line) until the application data has been changed on the database, that is, in most cases, until the R/3 update has been completed. This does not impair performance, since the lock is not a database lock.

The database lock (bottom dark blue line) is only set during the database LUW, in which the changes made in the R/3 System are actually updated (see [Functions of the Update Task \[Page 110\]](#)).

Details on the method of operation of the lock concept are provided under:

[The Owner Concept \[Page 81\]](#)

[The Lock Table \[Page 84\]](#)

[Lock Collisions \[Page 87\]](#)

[Cumulation of Locks \[Page 90\]](#)

Functions of the R/3 Lock Concept

[Questions and Answers Regarding Locks \[Page 92\]](#)

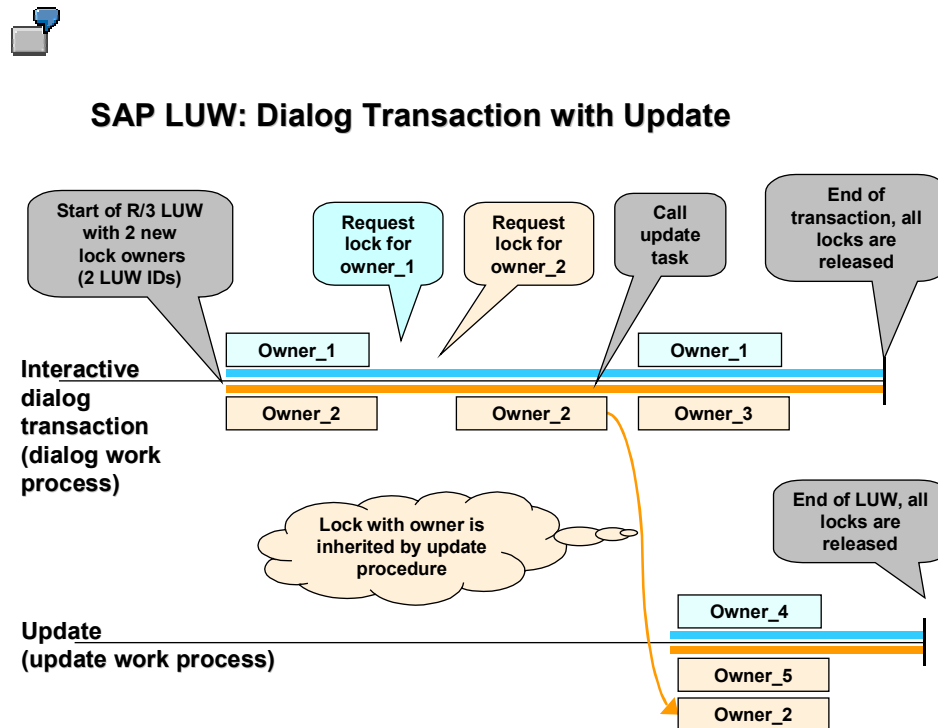
The Owner Concept

At the start of an R/3 transaction, two *owners* are always created and can request locks.

An owner is identified by his or her owner ID, as described in the section entitled [The Lock Table \[Page 84\]](#).

A lock can have one or two owners. This is determined by the ABAP programmer using the `_SCOPE` parameter (see [Function Modules for Lock Requests \[Ext.\]](#)).

The diagram below shows an example of an SAP LUW from a lock perspective.



At the start of the dialog transaction, the system creates two lock owners: the *dialog owner* Owner_1 and the *update owner* Owner_2.

During the course of the transaction, Owner_1 requests a lock, as does Owner_2 slightly later. When the update task is called (see also [Functions of the Update Task \[Page 110\]](#)), the lock and Owner_2 are inherited by the update task. An update work process is started with two owners, in the same way as a dialog work process, and then has three owners until the update is completed. All of the locks are released with an implicit `DEQUEUE_ALL` at the end of the update, at the latest.

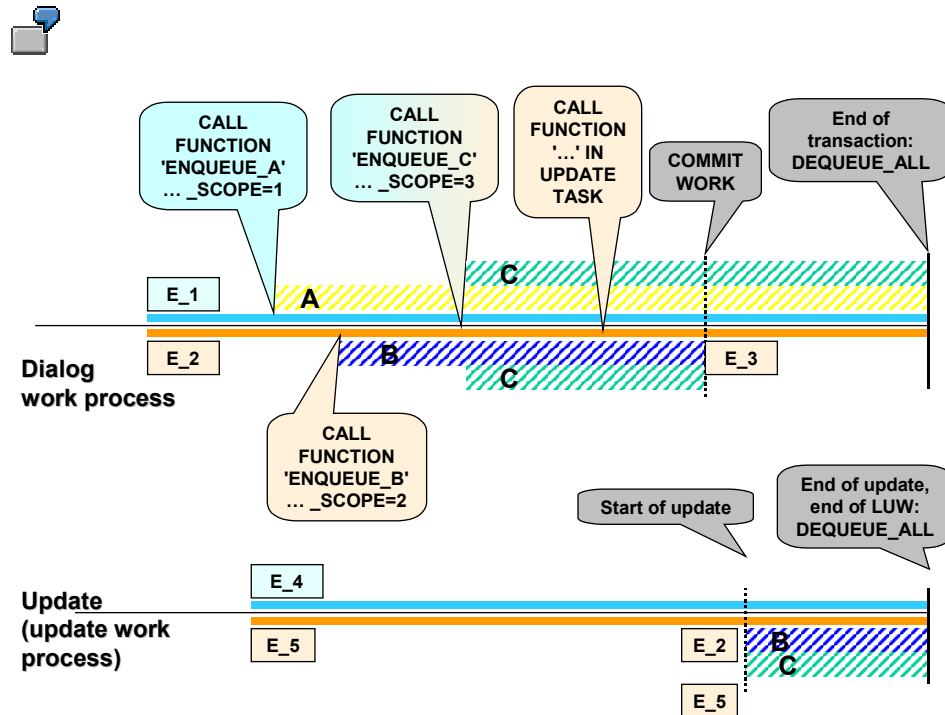
The following diagram shows the locks during the course of an SAP LUW in conjunction with the `_SCOPE` parameter. The application programmer uses this parameter to determine which of the two owners should actually own the lock. The diagram also shows how long the R/3 locks are active (these are not the actual database locks, as described under [Functions of the R/3 Lock Concept \[Page 77\]](#)).

`_SCOPE` can have the values 1, 2, or 3:

The Owner Concept

<code>_SCOPE = 1</code>	The lock belongs to Owner_1 only and, therefore, only exists in the dialog transaction.
<code>_SCOPE = 2</code>	The lock belongs to Owner_2 only and, therefore, is transferred to the update process
<code>_SCOPE = 3</code>	The lock belongs to Owner_1 and Owner_2.

The lock remains in place until either the relevant **DEQUEUE** function module is called or (as shown in the diagram) the transaction is completed with an implicit **DEQUEUE_ALL**.



During the course of the transaction in this example, the lock object A (which was [created \[Ext.\]](#) previously by the programmer in the ABAP Dictionary) is locked by the function call `CALL FUNCTION 'ENQUEUE_A'`. Since the `_SCOPE` parameter is set to 1, the lock A is not forwarded to the update task (it belongs to the dialog owner E_1 only) and, therefore, is released with the function call `DEQUEUE_A` or, at the latest, at the end of the dialog transaction.

Later, the lock B, which belongs to E_2 (the update owner) (`_SCOPE=2`), and the lock C, which has two owners (`_SCOPE=3`), are requested. The `CALL FUNCTION '...' IN UPDATE TASK` generates an [update request \[Page 112\]](#). At `COMMIT WORK`, the update task is called and inherits the locks and the update owner of locks B and C. These locks are released when the update is completed. Lock C of the dialog owner, however, may exist longer (depending on how the transaction is programmed).

Dialog locks A and C remain in place until the end of the dialog transaction.

See also:

[The Lock Table \[Page 84\]](#)

[Lock Collisions \[Page 87\]](#)

[Cumulation of Locks \[Page 90\]](#)

The Lock Table

The Lock Table

Definition

The lock table is a table in the main memory of the enqueue server that records the current locks in the system. For each elementary lock, the table specifies the owner, lock mode, name, and the fields in the locked table.

Use

The lock table is used to manage locks. Every time the enqueue server receives a lock request, the system checks the lock table to determine whether the request collides with an existing lock (see [Lock Collisions \[Page 87\]](#)). If this is the case, the request is rejected. Otherwise, the new lock is written to the lock table.

Structure

Each elementary lock corresponds to a data record in the lock table.

The structure of the lock entries is shown below.

Owner_1	Owner_2	Backup Id	Elementary Lock		
			Lock mode	Name	Argument
•Owner Id •Cumulation counter	•Owner Id •Cumulation counter	•Backup Id •Flag	•X, E oder S	•Name of the locked table	•Lock arguments
⋮	⋮	⋮			

The individual field have the following meaning:

Field	Contents and meaning
-------	----------------------

The Lock Table

Ow ner _1	Owner id and cumulation counter of owner_1: the ID contains the computer name, the work process, and a timestamp. It is also used to identify the SAP LUW. The cumulation counter specifies how often the owner has already set this elementary lock.	
Ow ner _2	The above applies here to owner_2	
Bac kup Id	Backup Id (index indicating where the lock entry is stored in the backup file) and backup flag (0 (no backup) or 1 (backup)).	
Ele me ntar y lock	Lock mode	S (Shared lock, read lock) E (Exclusive lock, write lock) X (eXclusive lock, extended write lock, cannot be cumulated)
	Name	Name of the database table in which fields are to be locked
	Argument	Locked fields in the database table (linked key fields, can also contain wildcards)

Integration

The lock entries can be viewed for diagnosis purposes. This is described in the section entitled [Managing Lock Entries \[Page 98\]](#).

The Lock Table

Lock Collisions

The check to determine whether a lock request collides with an existing lock is carried out in two steps: first, the system checks whether the lock request collides with an elementary lock in the [lock table \[Page 84\]](#). If this is the case, the system checks whether an *owner collision* exists. (The same owner can request a write lock more than once, for example. This is described under [Cumulation of Locks \[Page 90\]](#).)

If a collision exists, the user of the dialog transaction receives a message indicating that the requested object is currently locked by a different user. In the case of non-dialog processes (such as batch inputs), the lock request is resubmitted later.

Collisions Between Elementary Locks









Two elementary locks collide if all of the following conditions are fulfilled:

- The *name* of the elementary lock (table in which the lock is to be set) is the same
- The *lock argument* is the same, or more precisely: the letters in each position are identical (the wildcard symbol (@) is identical to all letters)
- At least one element lock does not have *lock mode S* (read lock)

The following graphic contains examples of collisions between a lock request (left) and an existing lock (right).



Elementary Lock Collisions – Examples

Elementary Lock Request				Current Lock			
Lock Mode	Name	Argument		Lock Mode	Name	Argument	
1) E	TAB1	ABCD	↔	E	TAB1	ABCD	       
2) E	TAB2	ABCD	↔	E	TAB1	ABCD	
3) S	TAB1	ABCD	↔	S	TAB1	ABCD	
4) E	TAB1	ABCD	↔	S	TAB1	ABCD	
5) E	TAB1	AB@@	↔	E	TAB1	ABCD	
6) E	TAB1	ABCD	↔	E	TAB1	AB@@	
7) E	TAB1	@@CD	↔	E	TAB1	AB@@	
8) E	TAB1	@@CDE	↔	E	TAB1	AB@@	

1. Write lock **ABCD** for table **TAB1** collides with the existing write lock **ABCD** for table **TAB1**

Lock Collisions

2. Write lock **ABCD** for table **TAB2** does not collide with the existing write lock **ABCD** for table **TAB1** and is accepted because the lock names are different
3. Read lock **ABCD** for table **TAB1** does not collide with the existing read lock **ABCD** for table **TAB1** and is accepted because both are read locks
4. Write lock **ABCD** for table **TAB1** collides with the existing read lock **ABCD** for table **TAB1**
5. Write lock **AB@@** for table **TAB1** collides with the existing write lock **ABCD** for table **TAB1** because @=C and @=D.
6. Write lock **ABCD** for table **TAB1** collides with the existing generic write lock **AB@@** for table **TAB1** because C=@ and D=@.
7. Write lock **@@CD** for table **TAB1** collides with the existing write lock **AB@@** for table **TAB1** because @=A and @=B and C=@ and D=@.
8. Write lock **@@CDE** for table **TAB1** does not collide with the existing write lock **AB@@** for table **TAB1** because the 5th letter in each case differs (E is not equal to _)

If the elementary locks do not collide, the lock request is added to the lock table as a new entry. If the elementary locks do collide, however, the system checks for an *owner collision* (described in the following section).

Owner Collision

If elementary locks collide, a decision is made whether to accept or reject the lock request based on the owner of the locks (see [The Owner Concept \[Page 81\]](#)).

An owner collision exists if one of the following conditions applies to an elementary lock collision:

- At least one owner is different
- The owners are identical but at least one lock has mode X (extended write lock, no cumulation, see also [Lock Mode \[Ext.\]](#))

The following graphic shows examples of these situations, where O_x is an owner different to O_1 and O_2.



Owner Collisions – Examples

	Elementary Lock Request				Current Lock			
	Lock Mode	Owner_1	Owner_2		Lock Mode	Owner_1	Owner_2	
1)	E	O_1	O_2	↔	E		O_2	✓
2)	E	O_1	O_2	↔	E		O_x	✖
3)	E	O_1	O_2	↔	E	O_x		✖
4)	E	O_1	O_3	↔	E	O_1	O_2	✖
5)	E	O_1	O_3	↔	E	O_1		✓
6)	X	O_1	O_2	↔	E	O_1	O_2	✖
7)	S	O_1	O_2	↔	X	O_1	O_2	✖
8)	E	O_1	O_2	↔	E	O_x		✖
9)	E	O_1	O_2	↔	E		O_1	✖

1. No owner is different, no lock in mode X => no collision
2. Owner_2 is different => collision
3. Owner_1 is different => collision
4. Owner_1 is the same but Owner_2 is different => collision
5. No owner is different, no lock in mode X => no collision
6. No owner is different, but the lock request has mode X => collision
7. No owner is different, but the existing lock has mode X => collision
8. Owner_1 is different => collision
9. Owner_2 is different => collision

An owner collision, therefore, implies an elementary lock collision. Lock requests are only rejected if an owner collision exists.

See also:

[Cumulation of Locks \[Page 90\]](#)

[The Owner Concept \[Page 81\]](#)

Cumulation of Locks

Cumulation of Locks

As described under [Lock Mode \[Ext.\]](#), there are three types of locks:

- **Shared** (read lock): several transactions can set a read lock simultaneously (to read data that is not changed)
- **Exclusive** (write lock): simultaneous read or write locks for this object are rejected; only the same owner (see [The Owner Concept \[Page 81\]](#)) can request the lock again. This is referred to as *cumulation*.
- **EXclusive**, lock that cannot be cumulated (extended write lock): this lock also can only be requested once by the same owner.

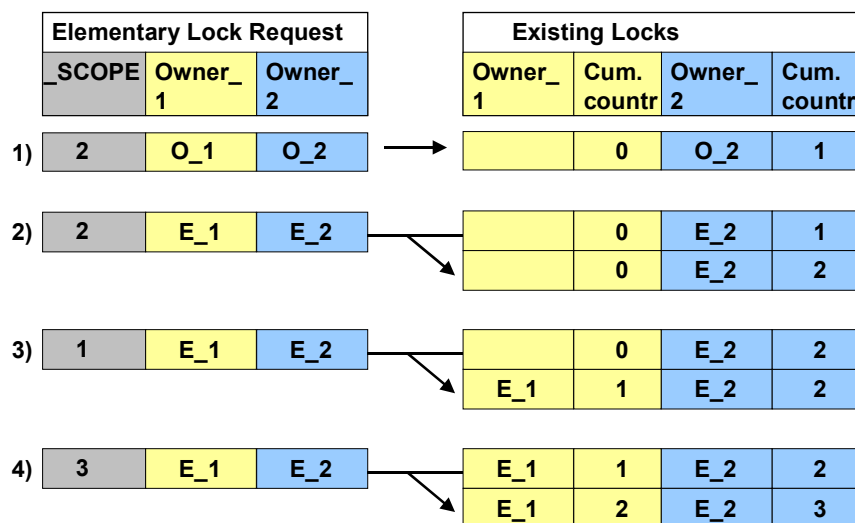
The type of lock selected is determined by the programmer of the transaction when the lock object is created. See also [The R/3 Lock Concept \[Ext.\]](#) in the ABAP Documentation.

A lock can be set more than once (cumulation), if the name, argument, and lock mode of the elementary lock are identical. The cumulation counter is incremented by one with each successive cumulation and reduced by one each time a lock is released. The lock is released when the counter reaches zero.

The following graphic illustrates how locks are cumulated.



Lock Cumulation – Example



1. The lock request is accepted and written to the [lock table \[Page 84\]](#). The cumulation counter of the dialog owner Owner_1 is 0. The counter of update owner Owner_2 is 1.
2. The lock request is also accepted. The cumulation counter of dialog owner Owner_1 is still 0 (because `_SCOPE` was set to 2); the counter of update owner Owner_2 is now 2.

Cumulation of Locks

3. This time, **_SCOPE** is set to 1. In other words, the cumulation counter of dialog owner Owner_1 is increased to 1 and that of update owner Owner_2 remains 2.
4. This time **_SCOPE=3**, that is, the lock has two owners. The cumulation counter of dialog owner Owner_1, therefore, is increased to 2 and that of Owner_2 is increased to 3.

Questions and Answers Regarding Locks

Questions and Answers Regarding Locks

The following section contains some common questions and answers regarding the lock concept and is intended to act as a quick reference guide. Most aspects are described in greater detail under [Functions of the R/3 Lock Concept \[Page 77\]](#).



Question

What happens to locks when the enqueue server is started?



Answer

If they have not been saved to disk in the backup file, they will be lost. Locks that have been transferred to the update task with **COMMIT WORK** after **CALL FUNCTION . . IN UPDATE TASK** are saved to disk. The locks are saved to disk when the update request becomes valid, that is, with the **COMMIT WORK**. Each time the enqueue server is restarted, the lock entries saved on the disk are reloaded to the [lock table \[Page 84\]](#).

A lock is saved to disk at the point at which the *backup flag* is set.



Question

Where is the lock table stored?



Answer

In the main memory (shared memory) of the enqueue server. All work processes on the enqueue server has access to the table. External application servers execute their lock operations in the enqueue process on the enqueue server. Communication in this case takes place via the relevant dispatchers and the message server.



Question

Can locks exist directly after startup?



Answer

Yes, the saved locks, which were inherited by the update task, are reloaded to the lock table during startup (see first question).



Question

How fast are lock operations?



Answer

In work processes on the enqueue server, a few 100 microseconds. In work processes on external application servers, the message server communication and 8 process changes are added to this. Depending on the CPU and network load, between approx. 20 (best case) and 80 (typical) milliseconds.

Questions and Answers Regarding Locks

**Question**

What should I do first if a problem arises?

**Answer**

Use the diagnosis functions:

sm12 Extras → *Diagnosis* and then

sm12 Extras → *Diagnosis in update*

If a problem is reported, back up the trace files `dev_w*`, `dev_disp`, `dev_eq*` and check the Syslog.

**Question**

The following message is displayed in the diagnosis details in SM12:

Lock management operation mode

Internal lock management in same process

What does this message mean and what are the other options?

**Answer**

"Internal lock management in same work process" in the diagnosis function means that you are logged onto the enqueue server and your work process can access the lock table straight away. You do not have to delegate enqueue requests to an enqueue process on a remote enqueue server. If you are logged onto an application server that is not an enqueue server, the diagnosis function will provide you with the name of the enqueue server.

Each R/3 System has exactly one application server that functions as an enqueue server. This enqueue server maintains the lock table, which is located in a shared memory segment. All of the work processes on the enqueue server can access the lock table. All work processes on other application servers delegate their enqueue requests to a special enqueue work process on the enqueue server.



This procedure is configured automatically. The parameter line `"rdisp/enqname =<application server name>"` in the default profile `DEFAULT.PFL` indicates which application server is currently acting as the enqueue server. When an application server detects that its name matches the name of the enqueue server, it creates the lock table and all of its work processes process enqueue requests inline. If an application server detects that its name does not match the name of the enqueue server, it sends all enqueue requests to the enqueue server.

Work processes of the type "enqueue" guarantee that incoming requests are processed immediately. One enqueue process is usually sufficient. In very large R/3 Systems with many application servers, a second process can be beneficial. However, it is not expedient to define more than two enqueue processes. If the transaction `SM50 -> [CPU]` shows that only the first enqueue process is being used, the bottleneck is due to something else.

Questions and Answers Regarding Locks



Question

Why is an enqueue work process required in a central system? Don't all work processes have the same access to the shared memory and thus to the lock table?



Answer

Although the enqueue process is not used in a central system, it does not do any harm. Since almost all customers install an application server sooner or later, problems will inevitably arise if the enqueue process is missing. For this reason, the enqueue diagnosis function will output an error if an enqueue process has not been configured.



Question

Are the locks in the lock table also set at the database level? If not, database functions could be used to process objects locked in R/3.



Answer

Locks are not set on the database. The lock table is stored in the main memory of the enqueue server.



Question

Is a lock table built if an enqueue work process is not started on the enqueue server in the instance profile?



Answer

Yes, because the work processes on the enqueue server process the lock table directly, and not via the enqueue process. The latter is only responsible for lock requests from external application servers.



Question

How can I find out who is currently holding the ungranted lock? In other words, how can check the program after an ENQUEUE to determine which user is currently holding the lock so that I can let him or her know?



Answer

When the ENQUEUE_... function module is returned, the name of the lock owner is listed in SY-MSGV1.



Question

*With a single-process system as an enqueue server, we have reached X SD Benchmark users. Can this number be increased by using a multiprocessor system (message server on the same machine as the enqueue server)? Can we assume that scaling is linear (number of CPUs * X SD users)? How many processes are advisable if message servers, dispatchers, one dialog, and two enqueue processes are to run on the system?*

Questions and Answers Regarding Locks

**Answer**

A significant increase in the enqueue server throughput can be expected by using several processors. The CPU load on the enqueue server is distributed relatively evenly between message servers, dispatchers, and enqueue work processes, which means that up to 3 processors can be occupied simultaneously. Dispatchers and message servers represent the bottleneck with the enqueue.

Linear scaling can be expected for up to 3 processors, even if lock requests are so frequent that message servers, dispatchers, and work processes are occupied simultaneously. Due to asynchronous system processes (for example, syncer), using more processors can further enhance throughput.

**Question**

The Syslog often contains messages such as "Enqueue: total wait time during locking: 2500 seconds". How should I analyze this problem? Or is the entry not critical? (There are no records of terminations or timeouts.)

**Answer**

The message is output for information purposes only but may indicate parallel processing errors with ABAP programs. The specified wait time is the time that has elapsed since startup due to the use of the WAIT parameter when the enqueue function module was called.

The WAIT parameter enables a lock attempt to be repeated a number of times, for example, so that the update task does not have to be cancelled when a lock is set temporarily by other programs. The work process remains busy between the lock attempts.


Important Profile Parameters for the Lock Concept

Important Profile Parameters for the Lock Concept

System profile parameters are used to adapt the system as best as possible to the current requirements and resources. A general description of these parameters is provided under [Profiles \[Ext.\]](#) in the CCMS documentation.

The most important profile parameters for updating are described under [Main System Profile Parameters for Updating \[Page 128\]](#).

The following table describes the main profile parameters associated with the lock mechanism. You can obtain a complete overview by searching for the parameters with the entry ***enq*** in transaction **rz11**.

Parameter	Meaning
enqueue/table_size	<p>Size of the lock table [Page 84] managed by the enqueue server in the main memory. The lock table contains information on which locks are currently held by whom.</p> <p>Lock table overflows are recorded in the system log (see System Logs [Page 44]).</p> <p>In this case, you should check whether the update server is functioning correctly, since the lock table can grow very fast if the update function stops. If no update problems exist, you can use this parameter to increase the size of the lock table.</p> <p>The Computing Center Management System (CCMS) monitors the status of the lock table constantly and outputs warnings if the space available is not adequate.</p> <p>You can check the space available in the lock table by displaying the lock statistics [Page 106].</p> <p>Unit: kilobyte</p> <p>Default value: 4096</p>
rdisp/wp_no_enq	<p>Number of enqueue work processes that are to run on this instance.</p> <p></p> <p>Naturally, this parameter is set to 0 on all application servers on which the enqueue server is not running, since only the instance with the enqueue server can contain enqueue work processes.</p> <p>Unit: integer</p> <p>Default value: 0</p> <p>The parameter on the enqueue instance is usually set to 1, that is, there is only one enqueue work process in the entire system. The parameter, however, can be set to 2-4 in large systems.</p>

Important Profile Parameters for the Lock Concept

enqueue/backup_file	<p>Complete path to the backup file. This is used to enter the locks, which are transferred to the update, in the lock table again after the enqueue server is restarted (see also Functions of the R/3 Lock Concept [Page 77]).</p> <p>Unit: character string (path name)</p> <p>Default value: /usr/sap/<SID>/D<instance no.>/log/ENQBCK</p>
rdisp/enqname	<p>Name of the application server that provides the enqueue service (see Functions of the R/3 Lock Concept [Page 77]).</p> <p>The name should be maintained in the default profile so that all application servers use the same name.</p> <p>Unit: character string (instance)</p> <p>Default value: name of the central instance</p>



These are only some of the profile parameters associated with lock management. The default values are designed to ensure optimum system performance and should only be changed by experts.

Managing Lock Entries

Use

The purpose of lock management is to monitor the lock logic of your system. You can determine the locks that are currently set. Locks for which the backup flag is set (because they have already been transferred to the update task) are highlighted (see [Choosing and Displaying Lock Entries \[Page 100\]](#)).

This enables you to detect and rectify problems, for example, by [deleting \[Page 103\]](#) locks that are no longer required.

Integration

The R/3 lock concept works in close conjunction with the R/3 update facility, which is described in the section entitled [Update in R/3 \[Page 109\]](#).

Features

The initial screen of the lock management facility is structured as follows:

Sperreinträge selektieren

Auflisten

Tabellen-Name

Sperr-Argument

Mandant 000

Benutzer-Name MUSTER

BIN (1) (000) is0201 INS

You can use the lock management facility to

- [Choose and display lock entries \[Page 100\]](#)
- [Delete lock entries \[Page 103\]](#)
- [Test the lock management facility \[Page 105\]](#)
- [Display the lock statistics \[Page 106\]](#)

Activities

To display the lock management facility, choose *Tools → Administration, Monitor → Lock entries* or enter transaction code **sm12**.

Choosing and Displaying Lock Entries

Choosing and Displaying Lock Entries

You can display lock entries according to the following criteria (see also [Functions of the R/3 Lock Concept \[Page 77\]](#)):

Sperreinträge selektieren

Auflisten

Tabellen-Name

Sperr-Argument

Mandant

Benutzer-Name

BIN (1) (000) ds0025 INS

- Table in which rows are locked
- Lock entry argument
- Client
- User who set the lock

The client and your user are added automatically. If you want to display the locks of a different user or client, you must overwrite the relevant field. If you leave the field blank or enter a *, the data is selected according to all possible entries.

The lock entry display contains the following information:

Choosing and Displaying Lock Entries

Man	Benutzer	Zeitpunkt	Shared	Tabelle	Sperrargument
000	BRAUCKHOFF	14:54:25		SCR_P_ENQ	1300SAPMSLIC
000	BAREISST	14:56:17		TRDIR	VERDATA0
000	FANH	14:59:43		SCR_P_ENQ	0100FANTEST_TABLECONTROL
000	BINDEWALD	15:21:54		TRDIR	RSSYSTDB
000	AGHADAVOODI	15:36:14		TFDIR	HTTP_TRFC_HANDLER
000	SCHIED	15:42:58		TRDIR	SCHIEDTMP
000	AGHADAVOODI	15:51:29		TRDIR	MATEST19
000	SANTA-CRUZ	15:56:04		DOKHL	RERSBDC_ARCHIVE
000	FUCHSS	15:59:05		TFDIR	RSPO_RCHANGE_SPOOLREQ_ATTR

Selektierte Sperreinträge: 9

- The **Man** column contains the client in which a lock entry was created.
- The **User** column contains the user who set the lock (that is, the user who executed the ABAP program that created the lock).
- The **Time** column shows when the lock entry was generated.
- In the **Shared** column, you can see whether a lock object is being used by several users. In this case, the column contains an X. Otherwise, it is empty.

If a lock object has more than one shared lock, several users can lock the same data at the same time. See also [Lock Mode \[Ext.\]](#) in the ABAP Dictionary documentation.

- The **Table** column contains the tables in which rows are locked.
- In the **Lock argument** column, you can see the argument (key field) of the lock entry.

This corresponds to the entries in the [lock table \[Page 84\]](#).

The lock entries are shown in different colors:

Blue means that the locks have already been transferred to the update task (see also [The Owner Concept \[Page 81\]](#)), with the result that the backup flag is set. These locks are also rewritten to the lock table when the enqueue server is restarted.

Black means that the lock (still) belongs to the dialog owner. The backup flag is not set.

Choosing and Displaying Lock Entries

By choosing *Edit* → *Sort by*, you can display the locks according to user, time, table, or host system (host).

By double-clicking a lock entry, you can display detailed information, including the host name and number of the R/3 System in which the lock was generated.

Deleting Lock Entries

Use

Certain problems can result in locks that cannot be released again, with the result that users cannot access the locked objects.

These locks, however, can be deleted manually.



In general, you should not delete lock entries directly with the deletion functions in lock management. Unreleased locks are almost always a symptom of a different problem. When you solve the problem, the lock entry will be released automatically.

See also: [Questions and Answers Regarding Locks \[Page 92\]](#)

Prerequisites

The most frequent causes of unreleased lock entries:

- **Update problems.** Locks set by the update function are maintained by the system until the database change has been processed or terminated prematurely. If a problem is preventing the update, the locks generated by the update records waiting to be processed block the system. (See also [Analyzing and Rectifying Update Errors \[Page 165\]](#).)

You can identify locks held by update records in the list of lock entries. The locks held by the V1 part of the update record are highlighted in the list. The backup flag is also set for these logs in the detailed display.

Solution: to determine whether an update problem exists, proceed as follows:

- a) Note the lock key of the relevant lock entry. Select this entry and choose *Details* to display the transaction code.
- b) Choose *Tools* → *Administration* → *Monitor* → *Update* ([Update Management \[Page 135\]](#)) to check whether any unprocessed update records (status `init` or `auto`) exist for the user. If you find any unprocessed update records, identify the transaction code and lock key to locate the update record that is holding the lock.

Note that update records with the status `err` do not hold locks. Locks are released automatically when an update task is terminated prematurely. If you find `err` update records, follow the procedure described under [Analyzing and Rectifying Update Errors \[Page 165\]](#).

- c) Check whether your update server is running correctly. Choose *Tools* → *Administration* → *Monitor* → *System monitoring* → *Servers* or transaction `sm51`. Position the cursor on the application server on which an update server is running and choose *Processes* to display the work processes on the update server. If open update records exist and the update is running normally, you should be able to see the update processes changing.

If necessary, start the update server by choosing *Tools* → *CCMS*.

- **Premature SAPGUI termination.** If a user switches off the PC without first logging off, or if the SAPGUI program is terminated prematurely for some reason, the user may remain

Deleting Lock Entries

logged onto the R/3 System. The locks held by the user when the problem occurred are not released because the user is no longer active in the R/3 System.

Solution: you can release these locks by logging off the user:

- a) Choose *Tools → Administration → Monitor → System monitoring → Servers* or transaction sm51.

Display the users of the SAP instance that are specified in the detailed display for the lock entries. Check when the user was last active in the system.

If the user was not active for a long time, he or she may have shut down the PC without completing his/her work or SAPGUI may have terminated prematurely.

If possible, contact the user directly. His or her work will be lost when you log him/her off.

- b) If the user's SAPGUI was terminated prematurely, you can log off the user in the user overview. The user cannot request the lost locks again, even if he or she logs on again. In this case, he or she must repeat the work carried out on the locked objects.
- c) If the user simply has not finished his/her work, he/she should be able to complete the work on the locked object normally. His or her work will be lost when you log him/her off.



Only delete lock entries if you are sure that they are invalid and can be deleted without impairing an active process (user session, update, and so on). Deleting a lock held by active users may result in data loss or inconsistencies.

Procedure

If you want to delete an individual lock entry, position your cursor on the entry and choose *Lock entry → Delete*.

By choosing *Lock entry → Delete all*, you can delete all the lock entries at once.

A warning is display first indicating the risks associated with deleting lock entries. Following this, a popup is displayed in which you can delete the lock entry.



You can deactivate the warning for the current transaction by choosing *Edit → Suppress warning*.

Result

The locks are released and can be requested again.

Testing Lock Management

You can use the diagnosis functions in the *Extras* menu to test the lock management facility for errors. Two test functions are available:

- *Diagnosis*: checks whether the lock server is responding to lock requests. When a user locks an object, the work process in which the user is active and the lock server communicate to enter locks and then release them again. See also [Functions of the R/3 Lock Concept \[Page 77\]](#).
- *Diagnosis in update*: this function checks whether a lock output for a transaction can be forwarded to an update server, which is active during the update, and released again.

The section entitled [Questions and Answers Regarding Locks \[Page 92\]](#) contains additional information on these diagnosis functions.

By choosing *Extras* → *Function info*, you can display a brief explanation of how the lock management facility is implemented in a distributed environment.

Displaying Lock Statistics

Displaying Lock Statistics

Use

You can use lock statistics to monitor the locks that are set in your system. This can be useful for rectifying configuration problems.

Prerequisites

You do not need to log onto the application server on which the lock server is running; the global statistics are always displayed.

Procedure

In the initial lock management screen, choose (**sm12**) *Extras* → *Statistics* to display the statistics. These are the statistics that have been compiled since the last time the lock server was restarted.

Result

The information provided by the enqueue statistics is explained in the table below.

Enqueue Statistics at 1999/02/16 09:53:50

Enqueue Requests.....: 110914	Number of lock requests
Rejects.....: 217	Rejected lock requests
Errors.....: 0	Errors
Dequeue Requests.....: 29149	Release requests (DEQUEUE)
Errors.....: 0	Release (dequeue) errors
DequeueAll Requests...: 224832	Release of all locks for a LUW
CleanUp Requests.....: 217	
Backup Requests.....: 732	
Reporting Requests...: 66	
Compress Requests....: 0	
Verify Requests.....: 0	
Writes to File.....: 12548	Write accesses to file
Backup.....: 7091	
Owner Names.....: 5457	Maximum number of owner names that can be stored in the lock table
Peak Util.....: 5	Maximum number of owners stored simultaneously in the lock table
Actual Util.....: 0	Current number of owners in the lock table
Granule Arguments.....: 5457	Similar: occupancy of the lock table, elementary locks - affecting arguments

Displaying Lock Statistics

Peak Util.....: 65	
Actual Util.....: 0	
Granule Entries.....: 5457	Similar: occupancy of the lock table, elementary locks - affecting names
Peak Util.....: 65	
Actual Util.....: 0	
Update Queue Peak.....: 2	
Actual: 0	
Total Lock Time.....: 2407.752370s total	Total time required for lock operations
Total Lock Wait Time.: 1066.962126s total	Total wait time



The **occupancy of the lock table** is important for monitoring the lock mechanism. This should never overflow. An overflow is pending if the `Peak Util` for `Owner Names`, `Granule Arguments`, or `Granule Entries` is equal to or has nearly reached the maximum value.

Analyzing and Rectifying Problems

Analyzing and Rectifying Problems

If you experience problems with lock management (Syslog entries, locks that cannot be released, and so on), you may find the solution in one of the following notes:

Note number	Short text
0193864	Enqueue, qRFC, ID is assigned several times
0190985	Enqueue Patch Collection 46B 1999/12
0190701	Core Dump in Dequeue Rel. 45B from Patch Level 299
0187787	Enqueue queue overflow
0170602	Error Message: LOG R1J=> ThAdXEnqState, reset mode <<user name.
0149166	Enqueue: GE9, Accumulated Wait Time for Lock
0143774	Lock Table Defective After Overflow
0138559	Display of Enqueue Logging File Incorrect
0138542	Enqueue Request Fails
0137500	Enqueue Does Not Work
0135971	Error in the Enqueue Reset
0120030	SM12 You Cannot Update Lock Entries
0119705	Message LstRestore: Restore From Replication Failed
0112075	Enqueue Work Process Core Dump
0110948	Transaction Reset After Loss of Lock Table
0109473	Exclusive Lock Waits on Table USRBF, Deadlock
0079084	Syslog: Error When Writing the Lock Handler File
0079001	Error When Reading a Lock Handler File
0072301	Syslog: GE1 T Enqueue: OpCode 2, missing par. USER
0065972	Lock Entries with Time 00/00/00 Cannot be Deleted
0043614	Enqueue, remaining lock entries
0013907	System Error in Lock Handler, Lock Table Overflow
0000651	Requirements for Enqueue Handling

Updates in the R/3 System (BC-CST-UP)

Purpose

In the R/3 System, a business process is mapped by means of an SAP transaction, which can include several screen changes (for example, creating an order). The changes to the data caused by this business process should be written to the database in full or not at all. If the operation is canceled while the transaction is being executed, or if an error occurs, the transaction should not make any changes to the database. These activities are handled by the SAP Update System, which is described below.

The Update System also enhances reliability and performance, and makes it easier to restore the data when changes are made to the database.

Integration

Updates are closely related to the R/3 lock concept, which is described in the documentation on [The R/3 Lock Concept \[Page 76\]](#).

You can find further documentation on updates in [Update Techniques \[Ext.\]](#) in the ABAP documentation. This documentation explains how an application programmer can control the update logic using ABAP statements.

Features

The Update System is used to lighten the workload of the SAP transactions when time-consuming changes are made to the database. These are carried out asynchronously in special update work processes. The Update System also avoids rollback problems that occur as a result of the differences between the logical units of work in an SAP transaction and in the database.

Exactly how an update is made in the R/3 System is described in the section entitled [Functional Description of Updates \[Page 110\]](#).

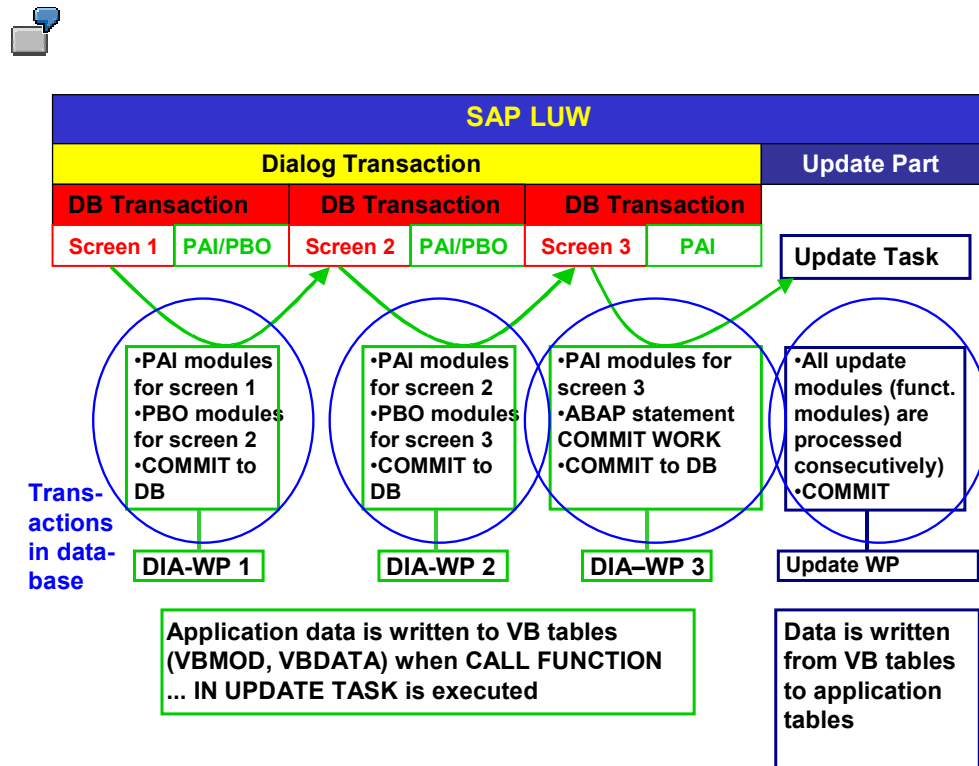
This documentation also contains sections on

- [Update Management \[Page 135\]](#) (transaction sm13)
- [Monitoring Updates \[Page 161\]](#)
- Analyzing and correcting [update errors \[Page 165\]](#)

Functional Description of Updates

Functional Description of Updates

The following graphic shows you how the R/3 System and the database interact when an SAP transaction is being executed, and illustrates the difference between a database LUW (Logical Unit of Work) and an SAP LUW.



The graphic shows an example of an **SAP LUW**. The dialog transaction extends over 3 screens. Each screen can be processed by a different (dialog) work process because while the system is waiting for the user to make an entry, the work process can be assigned a different task by the dispatcher.

The dialog transaction is completed with the statement COMMIT WORK; the update area of the LUW is then initiated: the update server (update task in the graphic) transfers the [update request \[Page 112\]](#) to an update work process. The time interval between two screen changes in the dialog transaction corresponds to a **database LUW** (completed with a COMMIT to DB); the update area is performed in a database LUW. The update area is described in greater detail in the section entitled [The Update Process \[Page 115\]](#).

Database LUW vs. SAP LUW

In the case of the **database**, an LUW is a **sequence of data operations that cannot be divided up**. The operations are either carried out in full or not at all. Database LUWs are modules which go to make up the database procedures for consistent data processing. An R/3 transaction can include several database LUWs (see example above), each of which can be terminated with a database COMMIT, which is generated automatically.

Functional Description of Updates

By way of contrast, an LUW for the **R/3 System** is a **business process, which cannot be divided up**. The process is either executed in full or not at all. **The SAP LUW of an R/3 transaction usually has to contain several database LUWs**. Under normal circumstances, it contains a dialog transaction (which maps a business process), and the command for writing the data to the database (update).

In each database LUW, data is written to the database to special **update tables** (and not to the application tables). Once the dialog area has been completed, all of the data in a **database LUW** is copied to the application tables: the [update request \[Page 112\]](#) is then performed. This is described in greater detail in the section entitled [The Update Process \[Page 115\]](#).

Advantages of the SAP Update System

The update system offers a number of advantages over database changes that are made directly in a transaction. These include:

- Enhanced performance for dialog transactions: time-consuming database changes are carried out asynchronously at the end of the LUW after the user has already moved onto the next LUW.
- Problems that arise as a result of different types of logical units of work (for database systems and for R/3 transactions) are avoided. By [bundling updates \[Page 118\]](#), the Update System supports critical rollback and restore functions for transactions.

To enhance performance even further, the application programmer can configure different types of updates (see [V1 and V2 Update Modules \[Page 124\]](#)):

- Time-critical V1 updates
- Non-time-critical V2 updates, which depend on V1 updates
- Non-time-critical V2 updates, which are gathered together and processed later ([collective runs \[Page 114\]](#))

See also:

Further details on the update process can be found in the following sections:

[The Update Process \[Page 115\]](#)

[Distributed Processing of Updates \[Page 123\]](#)

[Main System Profile Parameters for Updates \[Page 128\]](#)

[Reporting Update Problems \[Page 133\]](#)

Update Request

Update Request

Definition

An update request or *update record* describes the data changes defined in an SAP LUW, which are carried out either in full or not at all (in a database LUW). (This only applies to V1 updates. V2 updates are triggered once the V1 update has been completed, and therefore take place in a separate database LUW.) See also [The Update Process \[Page 115\]](#).

An update request is identified by means of its **update key**.

Use

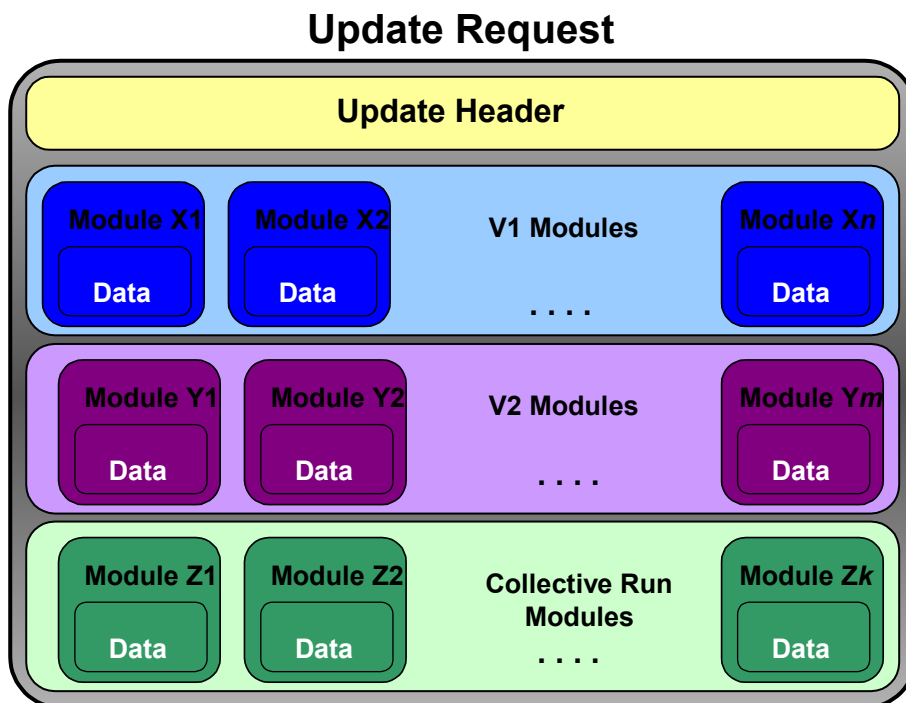
Once the dialog transaction has been completed (**COMMIT WORK**), and the update task has been called up, the update header is created. The update request is then created. The update data is stored in the update modules (function modules), which were created with the ABAP statement **CALL FUNCTION '...' IN UPDATE TASK**. The function module type is defined in the transaction for maintaining function modules (transaction **se37**, [Function Builder \[Ext.\]](#)). See below for further details.

Structure

An update request comprises an update header, V1 modules (or components), V2 modules and a [collective run \[Page 114\]](#).

An update module corresponds to a function module, and contains the update data and, in certain cases, error information, which is generated if the update is canceled.

The following graphic illustrates the structure of an update request.

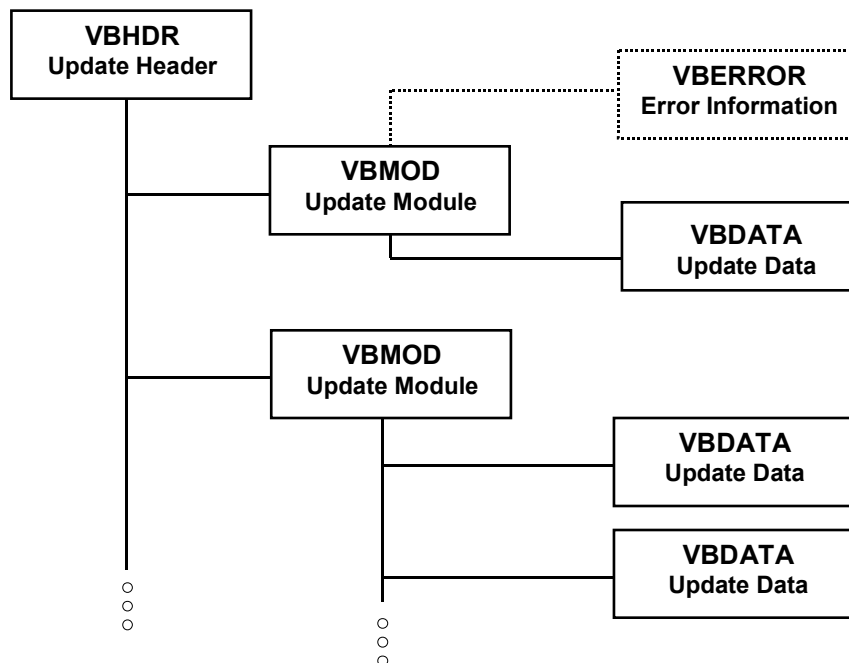


Integration

The following update tables in the database contain the following information:

Update Table	Contents/Description
VBHDR	Update headers (one per update request), see also Displaying the Update Header [Page 139]
VBMOD	Update modules (one per function module), n V1 modules and m V2 modules per update request (see also Displaying Update Modules [Page 141])
VBDATA	Data which is transferred to the modules (variables, structures, internal tables)
VBERROR	Error information which is generated if an update is canceled

The update tables are structured as follows:



You can also call this table up in the R/3 System with the table maintenance transaction (se16). Detailed information on maintaining tables can be found in the relevant documentation.

Collective Runs

Collective Runs

Definition

In addition to V1 and V2 function modules, the system also supports collective run function modules. In contrast to the other modules, these are not updated until a special report (report `RSM13005`) starts the update (in background mode); in other words, they are not updated automatically. All function module calls are then collected, aggregated (see example) and updated together. They are handled in the same way as V2 update modules.

Use

When you program a transaction, you use collective runs if a function module is called up on a regular basis, and you want to avoid a situation whereby each individual call causes the database to be updated.



Time-critical updates must not be carried out using collective runs. In this case, the collective run is not carried out until (long after) the time-critical V1 updates have been completed.



One of the function modules increments a statistical entry by one. This is called up 10 times during the course of the transaction. If you implement this as a V2 function module, the function module is updated 10 times after the V1 has been completed; this means that the database is updated 10 times.

If you use classify this as a collective run, you can perform the update at any time in one single operation: the program establishes that the statistical entry has to incremented 10 units and carries this out in a database operation.

Integration

The collective run modules can be viewed in [update management \[Page 135\]](#) until the report that triggers processing is started. Update records whose V1 and V2 modules have been processed correctly, but which still contain collective runs, are set to status `∇2` (see [The Most Important Update Statuses \[Page 120\]](#)).

The Update Process

Purpose

The Update System is used, for example, to lighten the workload of the R/3 transactions when time-consuming changes are made to the database. The changes are carried out asynchronously - usually with short delays in between - by special update work processes.

This is why the Update System is widely used in R/3 transactions (by almost every transaction that changes business data), although transactions can also change the data directly in the database.

Prerequisites

The application programmer decides whether, and if so, how the Update System is used for developing transactions. The different options open to the programmer are described in detail in the ABAP manual in the section entitled [Update Techniques \[Ext.\]](#).



When you are working with the SAP Update System, it is important to make sure that only **inserts**, **updates** and **deletes** are performed in the update. The necessary data (with **select** etc.) should, of course, be collected beforehand. Programming that has not been carefully thought through can result in poor system performance, and can, in certain cases, cause serious problems.

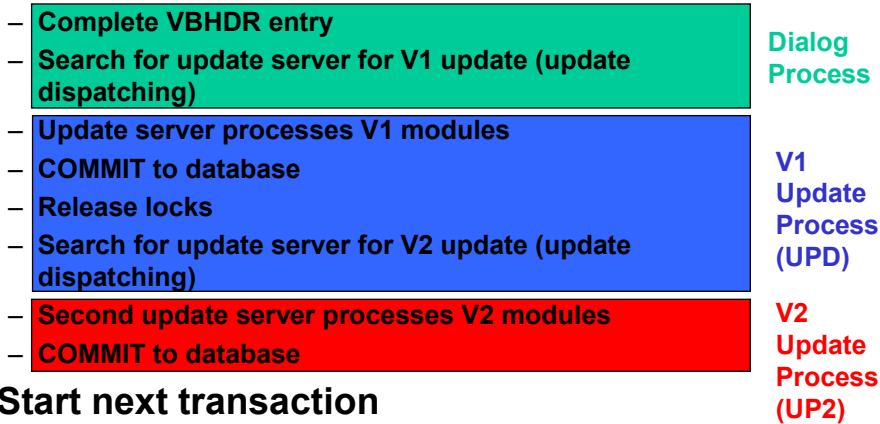
Process Flow

As illustrated in the graphic in [Functional Description of Updates \[Page 110\]](#), **COMMIT WORK** and the update task are called up at the end of a transaction; the dialog transaction comes to an end, and the update area of the SAP LUW is started. The following graphic illustrates the necessary actions and the sequence in which they carry out the different [work processes \[Ext.\]](#).

The Update Process

Update Procedure

- Complete transaction (COMMIT WORK)
- Call update task



- Start next transaction

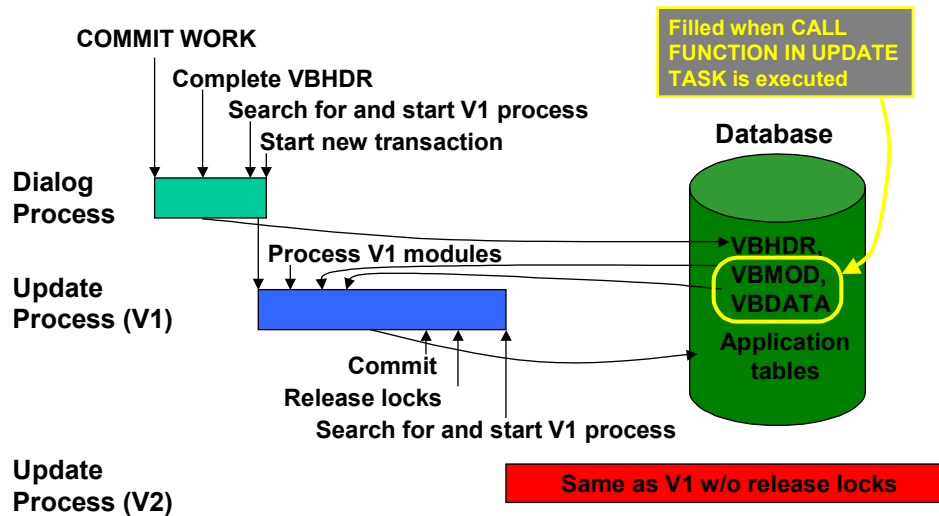
After the transaction has been processed, the dialog process completes the VBHDR entry (the update header of the [update request \[Page 112\]](#)) and searches for an update server for the V1 update. This process is described in greater detail in the section entitled [Update Dispatching with Load Balancing \[Page 125\]](#).

The update server distributes the tasks to an update work process. This processes the V1 modules of the update request, triggers a COMMIT to the database, and releases the R/3 locks on the update request (see [The R/3 Lock Concept \[Page 76\]](#)). The work process then searches for an update server for the V2 update, providing V2 update modules exist.

A V2 update server then passes this onto a V2 work process, which processes the V2 modules and triggers a COMMIT to the database.

The following graphic illustrates this from the point of view of the different work processes. The graphic also shows the times at which changes are made in the database.

Update Procedure – Work Processes Involved



Processing the V1 modules involves transferring the contents of the update tables VBMOD and VBDATA to the application tables of the database. The changes are not actually made to the tables in the database until the database LUW in which this takes place is completed. The R/3 locks are released and, if V2 update modules exist, the V2 update is started.

Result

If the update is completed without any errors, the update record is no longer listed in [update management \[Page 135\]](#) and the data is successfully processed.

Further information on the update process can be found in the following sections.

[Bundling Updates \[Page 118\]](#)

[Synchronous and Asynchronous Updates \[Page 119\]](#)

[The Most Important Update Statues \[Page 120\]](#)

[Error Handling and Data Consistency \[Page 122\]](#)

Bundling Updates

Bundling Updates

An R/3 transaction can, depending on how it is programmed, carry out database changes directly without using the SAP Update System; the database changes are made at the end of a database LUW.

The transaction can, however, also use the SAP Update System and bundle the updates of several database LUWs (see [Functional Description of Updates \[Page 110\]](#)).

Bundling is an ABAP programming technique which collects database changes and performs these together at the end of a transaction. (See also [Update Techniques \[Ext.\]](#).)



In the examples described so far, the updates have always been bundled.

Bundling updates involves combining important changes made by an R/3 transaction in a database LUW. Only changes that are performed in a single database LUW can be reversed. **If it is important to be able to reverse all of the changes made with an R/3 transaction for the sake of maintaining data consistency, all of the changes must be bundled in one database LUW.**

In order to bundle the updates, the special ABAP form routines and function modules for the database changes are not processed until the ABAP keyword `COMMIT WORK` is output.

The R/3 System also features its own locking mechanism for restricting data access. This mechanism is used, in certain cases, to hold R/3 locks within an R/3 transaction beyond the boundaries of database LUWs. (At the start of a new database LUW, the database locks of the previous LUW are released. The R/3 locks, however, remain active). Further information on R/3 locks can be found in the documentation on [The R/3 Lock Concept \[Page 76\]](#).

Update bundling and the R/3 lock system maintain data integrity in processes that cover several database LUWs, and fulfill the requirements regarding rollback. This means that all of the data changes can be reversed if a runtime error occurs during the update.

Further information on LUWs and bundling can be found in the ABAP manual.

Synchronous and Asynchronous Updates

Database changes that are made via the SAP Update System and transferred to an **update work process**, can be carried out synchronously or asynchronously. The mode is specified in the ABAP source code of the R/3 transactions and cannot be changed dynamically by the user.

The R/3 transaction creates an [update request \[Page 112\]](#) with `CALL FUNCTION ... IN UPDATE TASK`, and transfers this to an **update work process**. Data is then written to the update tables at the end of a database LUW.

With **synchronous updates**, the program that outputs the statement `COMMIT WORK AND WAIT` waits until the update work process outputs the status of the update.



Updates generated by batch input sessions are always carried out synchronously. Batch input with `CALL TRANSACTION USING` can be updated synchronously and asynchronously. In addition to this, every 100th update in a background job is carried out synchronously in order to avoid delays in the Update System.

With **asynchronous updates**, the program that outputs the statement `COMMIT WORK` passes the update onto the Update System and does not wait for the update process to respond.



Most of the updates in R/3 application transactions are programmed for the asynchronous mode. **These asynchronous updates are the objects that appear in update management.**

See also: [Update Techniques \[Ext.\]](#) in the ABAP manual.

The Most Important Update Statuses

The Most Important Update Statuses

Update management supports different statuses for update requests; these statuses are displayed in the update record overview (transaction **sm13**) in the **Status** field group.

The status indicates the phase of the [update process \[Page 115\]](#) that the request has reached, or in which the request has become "stuck".

The most important statuses are described below.

As already discussed in the section entitled [The Update Process \[Page 115\]](#), the dialog work process passes the update request onto an update work process after the dialog area has been completed. This then processes the V1 update modules. When the ABAP statement **COMMIT WORK** is received, the data is written to the database and the V2 update is output to a V2 work process (providing V2 modules exist in the update request).

The following statuses are possible during this phase:

Status	Phase
init	From the point at which the dialog work process passes the update request onto the update work process up until the COMMIT in the update work process.
err	If an error occurs in the init phase which prevents the update from being carried out.
V1	If the init phase is completed successfully, and the V2 modules are passed on for further processing. If no V2 modules exist, this update request no longer appears in the overview.
V2	If the V2 modules are also processed correctly, but there is still a collective run (can be regarded as V3) to be carried out. If there is no collective run [Page 114] to be carried out, this update request no longer appears in the overview.
ok	If the parameter rdisp/vb delete after execution [Page 128] is set to 2 - in other words, automatic deletion is deactivated - an update that has been completed successfully has the status ok . If automatic deletion is activated (default), the update record no longer appears in the overview.

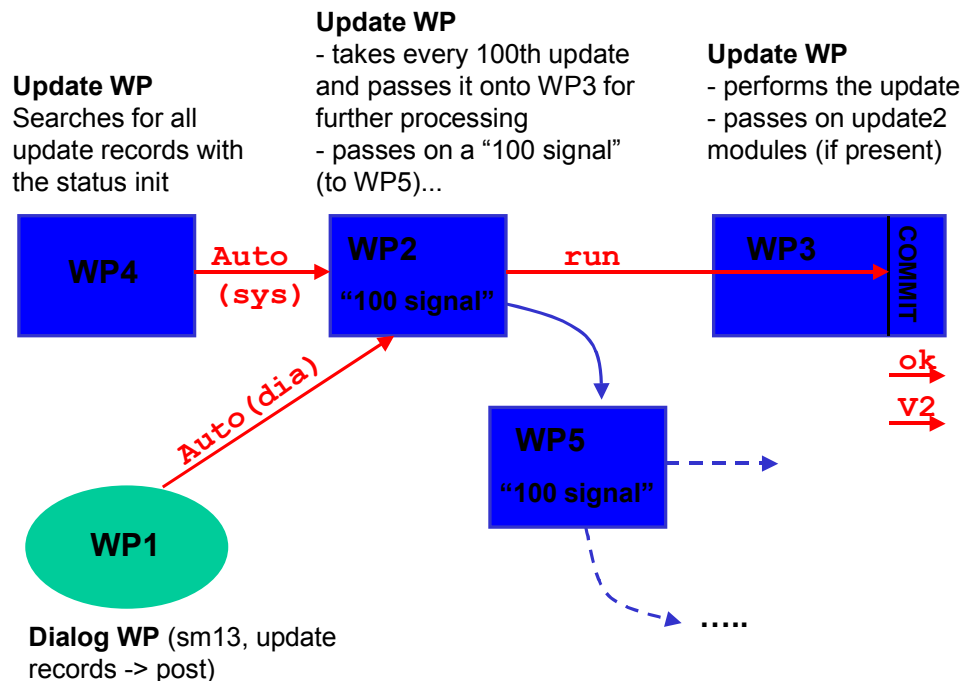
A situation may arise, however, in which an update record becomes "stuck" in the status **init** without switching to the error status **err**. If the record remains set to the status **init** for a prolonged period of time, the record can be updated subsequently in the following ways; the following statuses are then active (see also graphic).

Status	Phase
auto(dia)	The update record has been processed manually [Page 144] by the system administrator with transaction sm13 (<i>Update records</i> → <i>Update</i>). The dialog work process (WP1) transfers all of these update requests to an update work process (WP2) during which the update record is set to the status auto(dia) .

The Most Important Update Statuses

run	The work process WP2 collects the requests and passes them on in batches to a further update work process (WP3), which then performs the actual update. The record is set to the status <code>run</code> up until COMMIT in WP3.
auto(sys)	Each time an update server is restarted, this checks to determine whether the update requests are set to <code>init</code> . In this case, it initiates automatic processing of the requests by means of update work processes. This takes place in the same way as when the update is started manually, except that an update work process (WP4) starts everything and not just a dialog work process. The update record is then set to the status <code>auto(sys)</code> .

This is illustrated in the following graphic.



If, under exceptional circumstances, the update is not successful the first time, the status `run` corresponds to the status `init` when the update is repeated. If the update becomes "stuck" in the status `auto` or `run`, the status must be [reset \[Page 146\]](#) as only records with the status `init` can be entered with the methods described.

See also:

[Monitoring Updates \[Page 161\]](#)

[Analyzing and Correcting Update Errors \[Page 165\]](#)

Error Handling and Data Consistency

In order to ensure the consistency of the data, an update must be carried out in full or not at all. This is referred to as a **rollback requirement**. If a runtime error occurs in part of the update, all of the critical database changes made as a result of the update must be reversed. Less critical changes cannot be made.

With [update bundling \[Page 118\]](#) and the R/3 lock system, the R/3 Update System ensures that the data remains consistent (see also [The Update Process \[Page 115\]](#)).

The following section describes how the Update System responds if part of the update (V1 or V2) is canceled due to a runtime error. Various scenarios are possible here.

Cancelation in a function module of the update task V1 (IN UPDATE TASK requested)

- Updates already made for V1 functions are reversed.
- All other update task requests (V1, V2 or collective run) are discarded.
- The user receives an express mail indicating that the update was unsuccessful (see parameter [rdisp/vbmail \[Page 128\]](#))

Cancelation in a Function Module of the Update Task V2 (IN UPDATE TASK requested)

- Updates already made for V2 functions are reversed.
- Updates already made for V1 functions are not reversed.
- An error message appears on the screen if the system has been configured for this.

Cancelation in a Collective Run Function Module

- Collective runs are handled in the same way as V2 update modules.

Distributed Processing of Updates

The task of distributing the [update requests \[Page 112\]](#) (uniformly) among the different update work processes is performed by an **update server**.

The update server, therefore, acts as an intermediary which receives the update request and distributes it to the update work processes.



In R/3 System in which the code pages of the R/3 application servers have different character sets), the update work processes change their codes pages dynamically in line with the code page of the update request to be processed.

Update distribution can be deactivate and configured with parameters of the R/3 System profile. Further information can be found in the section entitled [Main System Profile Parameters for Updates \[Page 128\]](#) or in the update profile parameters in transaction rz11 (use the search string rdisp/vb* in rz11 to search for the parameters).

Types of Update Work Process

There are two types of update work process, one for V1 updates (listed as *Update* processes in transaction sm50/sm51 display) and one for V2 updates (listed as *Upd2* in sm50/sm51 display).



There must be at least one V1 update process in the R/3 System; there can, however, be more. If there is more than one process, update processing is distributed among the work processes on the basis of a load balancing algorithm (see also [Update Dispatching with Load Balancing \[Page 125\]](#)).

V2 processes are optional. These are used to avoid performance problems with time-critical V1 updates by splitting V2 update component processing. (V2 update components usually contain statistical data and are, therefore, less time critical). Further information can be found in the section entitled [The Update Process \[Page 115\]](#) and [V1 and V2 Update Modules \[Page 124\]](#). If there is no V2 work process, both types of update component are processed in the V1 work process. V2 work processes are also used if [collective runs \[Page 114\]](#) are carried out.

V1 and V2 Update Modules

V1 and V2 Update Modules

An update is divided into different modules (see also [Update Request \[Page 112\]](#)). Each module corresponds to an update function module.

There are two types of module.

The R/3 System makes a distinction between **primary**, time-critical (**V1**) and **secondary**, non-time-critical (**V2**) update modules. The system also supports [Collective Runs \[Page 114\]](#) for function modules that are used on a regular basis.

This distinction allows the system to process critical database changes before less critical changes.

- **V1 modules** describe critical or **primary changes**; these affect objects that have a **controlling function** in the R/3 System, for example order creation or changes to material stock.
- **V2 modules** describe less critical **secondary changes**. These are pure statistical updates, for example, such as result calculations.

The V1 modules are processed consecutively in a single update work process on the same application server. This means that they belong to the same database LUW and can be reversed. Furthermore, V1 updates are carried out under the R/3 locks of the transaction that creates the update (see [The R/3 Lock Concept \[Page 76\]](#)). This ensures that the data remains consistent; simultaneous changes to the objects to be updated are not possible.

V2 updates are carried out in a separate LUW and *not under the locks of the transaction that creates them*. If your R/3 System contains a work process for V2 updates, these are only carried out in this system. If this is not the case, the V2 components are processed by a V1 update process.

All V1 modules of an update must be processed before the V2 modules.



Let us assume that a transaction makes planning changes to a material and balance sheet, and updates two sets of statistics.

Each of these changes is represented by means of an update module (call update function module) in the [update request \[Page 112\]](#) – the two planning changes by a V1 update module (time critical), and the statistical changes by a V2 update module (less critical). (The V1 modules have priority, although the V2 modules are usually also processed straight away).

This is described in greater detail in the section entitled [The Update Process \[Page 115\]](#).

Update Dispatching with Load Balancing

Several **update servers** can offer update services in an R/3 System. In this case, the R/3 System automatically distributes the [update requests \[Page 112\]](#) among the available update servers using a process referred to as **update dispatching**.

Update processing is triggered by a request from the dialog server on which the update was generated. This request is sent to a specific update server. The dialog server thus decides which update server is to be used for a particular update.

Load balancing is based on a list of the available update servers. This list is stored on each application server and is automatically updated by the message server when a server is started or stopped, or when its configuration changes as a result of a new operation mode being set. You can display this list by choosing *Goto* → *Server*.

If there is more than one possible update server for processing an update record, the dialog server selects the server to be used on the basis of a load balancing algorithm.

This load balancing algorithm assigns update records to the update servers cyclically. A server sends update requests consecutively to each update server. If the number of requests sent is the same as the number of update work processes on an update server, the server is removed from the list until the start of the next cycle. Once all of the update work processes have been assigned update requests in this way in the R/3 System, the cycle starts again and with all of the update servers.

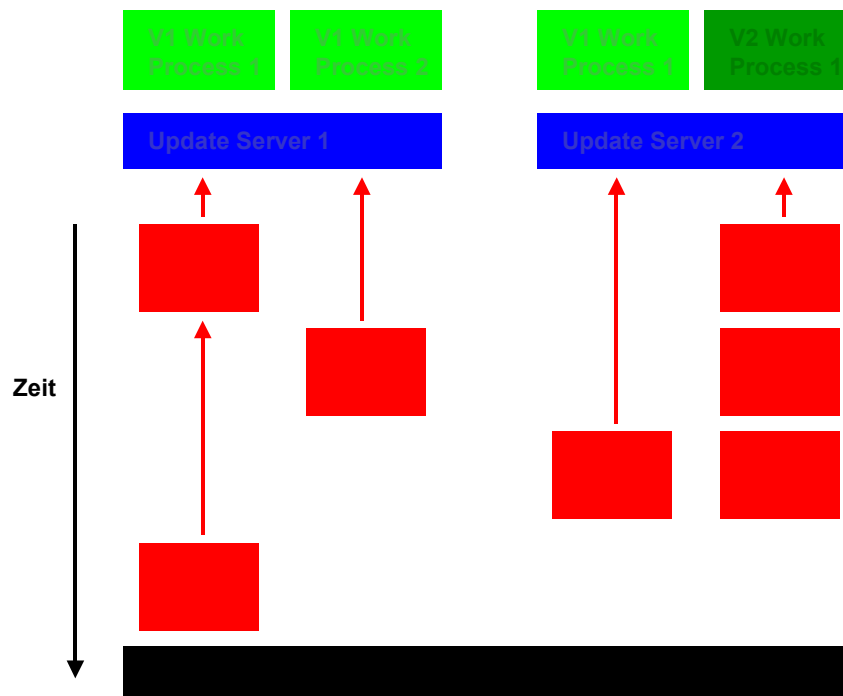
If a V2 work process *Upd2* has been defined, V2 update modules can only be processed by this work process.

If there are no V2 work processes, V2 updates are processed in V1 work processes. A V2 component must, however, wait until all of the V1 update requests have been processed.

The following example illustrates the load balancing algorithm.



Update Dispatching with Load Balancing



Two work processes run on update server 1 for V1 update modules. Update server 2 has one V1 and one V2 update work process. A load balancing cycle for V1 updates, therefore, comprises three update requests. These are distributed as follows:

1. The first and second update request are sent to update server 1 (number of requests = number of V1 update work processes * `rdisp/vb_factor` parameter (default value 1; see also [Additional System Profile Parameters \[Page 131\]](#)). Setting the parameter to a value other than 1 is advantageous if an update server is significantly faster and is, therefore, to be assigned more update requests.
2. The Update System switches to update server 2 for the third request. The load balancing cycle is then complete (number of assigned updates = number of update work processes).
3. V1 update request 4 is sent to update server 1 again and the cycle is repeated.
4. V2 updates are processed consecutively by the V2 work process.

Reassignment in the Event of an Update Server Failure

If, for some reason, an update server fails, the other update servers in the system perform any updates that the defective server was not able to handle.

If an update server is shut down, for example, the R/3 message server establishes that the server is not available on the basis of an error message regarding the missing server connection or on account of the connection checks, which are carried out at regular intervals.

Once the message server has been informed of this, it updates the list of application servers and associated services, and distributes the list to the other application servers.

The first update server on the list automatically assumes responsibility for distributing the update requests of the defective server on the basis of the dynamic update assignment criteria described above.

Main System Profile Parameters for Updates

Main System Profile Parameters for Updates

The following table contains an overview of the most important update parameters.



Note that you normally never have to set these parameters directly. Generally speaking, the most appropriate value is the default parameter value; otherwise the parameters are set for you by the profile management function (for example, if you set the number of update work processes in an instance profile).

Parameter	Description
rdisp/vb_dispatching	Activates (1) or deactivates (0) update dispatching with load balancing [Page 125] . Load balancing is normally active. Do not change this value unless instructed to do so by SAP.
rdisp/vbname	Specifies the name of the update server that is to process the updates if load balancing is deactivated (rdisp/vb_dispatching = 0). In the standard system, this parameter specifies the name of an update server (set when the update server is created). If rdisp/vb_dispatching is set to 0, the updates are only processed by the server in rdisp/vbname. If the parameter has not been set correctly (not to an update server), this is reported when the installation check is performed (transaction <code>sick</code>).
rdisp/vb_delete_after_execution	Determines whether update records are deleted automatically after they have been processed successfully. In the standard system, this is set to 1 (automatic deletion activated). If set to 2, automatic deletion is deactivated. This value can be used to set the update and database performance. In this case, the report <code>rsm13002</code> with the parameter <code>DELETE = X</code> should run in the background at least once a day to prevent the update tables from becoming excessively large. See also the section entitled Background Processing [Ext.] in the CCMS documentation.
rdisp/max_vb_server	Maximum number of update servers permitted in the R/3 System. Default = 50 servers.
rdisp/vb_included_server	List of the R/3 update servers, which are to be used to process updates in accordance with the load balancing principle. No updates are assigned to update servers that do not appear in the list. This parameter is empty in the standard system. This means that all active update servers are taken into consideration for the load balancing mechanism. This is generally speaking the optimum value.

Main System Profile Parameters for Updates

rdisp/vbdelete	<p>Specifies the number of days after which the update records are deleted. The parameter is set to 50 days in the standard system.</p> <p>Once this interval has expired, an update record is deleted irrespective of its status (processed, not processed, error etc.).</p> <p>If set to 0, automatic deletion is deactivated. This value should only be set temporarily, and only if an incorrect update record is to be kept for further analysis.</p>
rdisp/vbmail	<p>Specifies whether a user is to be informed via express mail that his update has been canceled.</p> <p>The parameter is activated (value = 1) in the standard system. If an update is terminated prematurely, the system sends an express mail via R/3 Mail to the user that created the update.</p> <p>If set to 0, the express mail is not sent automatically.</p>
rdisp/vbreorg	<p>Specifies whether an update server is to search for incomplete update records and delete these. Update records can be incomplete - if a transaction registers one or more update components - as a result of a screen change in which the transaction that creates the update generates a database commit; the transaction, however, is terminated prematurely and a rollback is triggered.</p> <p>Incomplete update records are of no significance, but take up space in the database. Reorganization is, therefore, active (value = 1) in the standard system.</p> <p>You can deactivate reorganization by setting the value to 0. In this case, you should make sure that the report rsm13002 is run in the background at regular intervals to delete the incomplete update records (see also Background Processing [Ext.]).</p>
rdisp/vbstart	<p>Specifies whether update requests that have not been processed or not fully processed (status = <i>run</i>, V2 modules not processed, etc.) are processed automatically when an update server is started. In the standard system (value = 1), update records such as these are processed (status usually = <i>auto</i>) (see The Most Important Update Statuses [Page 120]).</p> <p>If set to 0, automatic processing is deactivated. The value of this parameter can normally only be changed by SAP.</p>




To obtain further information on the update parameters, enter the transaction code /nrz11 in the R/3 System and then search for the parameters whose names start with rdisp/vb.

Main System Profile Parameters for Updates

Additional System Profile Parameters

The most important system profile parameters for updates are described in the section entitled [Main System Profile Parameters for Updates \[Page 128\]](#). The system also supports additional parameters, which can be useful for correcting errors and optimizing update performance. These parameters are described below.

You can access further detailed information if you call up transaction **rz11**, search for the parameters and then display the documentation.

Parameter	Description
rdisp/vb_context	<p>You can use this parameter to specify how V1 update dispatching [Page 125] takes place. The parameter is interpreted as a bit mask:</p> <p>Bit 0: the DB group is to be retained during dispatching (see also Configuring Update Groups [Page 149])</p> <p>Bit 1: an update server on which the current language is installed must be selected for dispatching.</p> <p>The default value 3 (both bits set) specifies, for example, that an update server is selected for update dispatching, which is in the same DB group, and on which the current language is installed.</p> <p>Changes should only be made in order to correct errors.</p> <p>Value range: 0-3</p>
rdisp/vb2_context	<p>This parameter controls update dispatching for V2 updates (see rdisp/vb_context).</p>
rdisp/vb_factor	<p>Update request dispatching takes place in accordance with a simple procedure, which is described in Update Dispatching with Load Balancing [Page 125].</p> <p>The parameter <code>rdisp/vb_factor</code> can be used to influence the number of update work processes relevant for dispatching in order to assign a greater number of update requests to servers that are particularly fast. The number of requests relevant for dispatching is the product of the current number of update processes and the value of <code>rdisp/vb_factor</code>.</p> <p>After the V1 part of an update request has been processed the V2 part is dispatched in the same way.</p> <p>The parameter value must have the following syntax:</p> <pre>rdisp/vb_factor = S=<instance_name>,F=<factor>; ... S=<instance_name>,F=<factor>;</pre>  <pre>rdisp/vb_factor = S=host1_C11_00,F=2.0; S=host2_C11_00, F=2.5;</pre>

Additional System Profile Parameters

rdisp/vb_key_comp	<p>You can use the parameter rdisp/vb_key_comp to change the structure of the update key (see Displaying the Update Header [Page 139]). The key usually comprises the date, time, microseconds, host name, system number and work process number.</p> <p>In order to structure the index more favorably, you can move variable components to the start of the key by setting the parameter accordingly.</p>
rdisp/vb_refresh_info	<p>This parameter defines the time interval after which the context of an update server is redetermined. The context includes the number of server update work processes as well as the code page and DB node of the server.</p> <p>The context of an update server can change, for example, as a result of a new operation mode being set. This is why the context is reread every hour.</p>
rdisp/vb_v2_start	<p>This parameter specifies whether the V2 phase of an update request is started automatically at the end of the V1 phase or by a background job after a delay.</p> <p>Value range:</p> <ul style="list-style-type: none">1: The V2 phase is started immediately after the V1 phase2: The V2 phase is not started automatically

Reporting Update Problems

The Update System draws your attention to update problems as follows:

- The graphical [Alert Monitor \[Ext.\]](#) reports all update problems automatically.
- The Update System sends an express mail if an update is terminated prematurely. The user is informed that an express mail has been received by a dialog popup.
- The user whose update was terminated prematurely receives an error message.
- A system-wide message is displayed if the update is deactivated automatically or manually.
- All terminated updates are maintained in [update management \[Page 135\]](#). They are assigned the status *err*. See also [The Most Important Update Statuses \[Page 120\]](#)

Express Mails and Error Messages

All users whose updates have been terminated prematurely are notified by the R/3 System of the update error not only by alerts but also by means of an express mail. The mail is sent in the R/3 System in which the problem occurred.

In the standard system, warnings per mail are activated; you can, however, deactivate them with the parameter *rdisp/vbmail* in the system profile (further information on this parameter can be found in the section entitled [Main System Profile Parameters for Updates \[Page 128\]](#) and in transaction rz11).

The R/3 System also displays a message pointing out the update error on the terminal of the user. Exceptions to this include situations in which a remote update is made by means of an RFC, or if the user logged off before the update was processed.

Note on Deactivating Updates

In the event of serious database errors, the update is stopped automatically (further information on this can be found in the section entitled [Automatic Update Stop in the event of Database Problems \[Page 134\]](#) and [Deactivating and Reactivating Updates \[Page 164\]](#)).

In these cases, you and all other users receive a warning message in the status bar or the active mode. All active transactions in the system are interrupted if updates are deactivated.

See also:

[Monitoring Updates \[Page 161\]](#)

[Analyzing and Correcting Update Errors \[Page 165\]](#)

Automatic Update Stop in the event of Database Problems

Automatic Update Stop in the event of Database Problems

If a serious error occurs in the database of the R/3 System, the update is stopped automatically. Each database error message which requires the intervention of a database administrator is also passed onto the Update System, which then interrupts the update.

The automatic stop mechanism is valid for all databases supported by the R/3 System.



Let us assume that your Oracle database has just output the message "Table space overflow". The R/3 database interface recognizes the message as a serious error message and outputs a signal to the Update System, which then stops the update. The active transactions are interrupted (an appropriate message is output here) until the update is reactivated.

Stopping the update in the event of database problems makes it easier to restore normal operating conditions after the error has been corrected. Updates are not canceled but are assigned the status *init* or *auto*, which indicates that they have not yet been completed. The updates are then processed automatically when the update is reactivated after the error has been corrected.

The update is not stopped if a local error in an update function module causes an update to be terminated prematurely (see also [Error Handling and Data Consistency \[Page 122\]](#)).

Further information can be found in the section entitled [Deactivating and Reactivating Updates \[Page 164\]](#).

Update Management

Use

Update management is used to

- display updates
- test and debug canceled updates
- reset updates
- delete updates
- display statistics on updates

The update management initial screen (transaction sm13) can be seen below:

Verbuchungssätze: Einstieg

Verbuchungssätze Bearbeiten Springen System Hilfe

Mandant 000
Benutzer HAUGT

Status
☐ Abgebrochen
☐ Noch zu verbuchen
☐ V1 ausgeführt
☐ V2 ausgeführt
☒ Alle

Die Verbuchung ist aktiv Info

Selektion
Ab Datum 15.10.1998
Ab Zeit 00:00:00
Max Anzahl Sätze 99999
Server

Integration

The sections entitled [Monitoring Updates \[Page 161\]](#) and [Analyzing and Correcting Update Errors \[Page 165\]](#) describe the procedures for monitoring updates and handling update errors.

Features

Update management supports the following functions:

[Selecting and displaying updates \[Page 137\]](#)

[Analyzing Canceled Updates \[Page 142\]](#)

[Testing Canceled Updates \[Page 143\]](#)

Update Management

[Processing Updates Manually \[Page 144\]](#)

[Resetting Update Statuses \[Page 146\]](#)

[Deleting Update Records \[Page 147\]](#)

[Reorganizing Update Records \[Page 148\]](#)

[Configuring Update Groups \[Page 149\]](#)

[Displaying and Resetting Update Statistics \[Page 152\]](#)

[Displaying Servers \[Page 154\]](#)

Many of these functions can be carried out more easily in [Update Program Administration \(Transaction sm14\) \[Page 155\]](#).

Activities

You can call up Update Management from the initial R/3 screen either by choosing *Tools* → *Administration* → *Monitor* → *Update* or - from any R/3 screen - by entering **/nsm13** in the command field.

Selecting and Displaying Updates

You can use the following criteria for selecting updates:

Selection options for update records

Option	Description	Default setting
Client	Client in which the update was created	Current client
User	User who created the update	Current user
Type of update record	<p>The following status selection options are available:</p> <ul style="list-style-type: none"> <i>Terminated</i>: only display canceled updates of the user <i>To be updated</i>: display updates of the user that have not yet been processed (the reason for this is indicated in the display; see below) <i>V1 executed</i>: display executed V1 updates of the user <i>V2 executed</i>: display executed V2 updates of the user <i>All</i>: display all update records of the user 	All
Date and time	Date and time as of which the updates are to be displayed	Today, 0.00 hours
Max. no. records	Maximum number of update records that can be displayed	99999
Server	Update server on which the update was carried out	



If you leave a field blank or enter *, a generic search is carried out for all possible entries.

Displaying Update Records

This function allows you to display all of the update records that not yet been fully processed.

The selected data records display contains the following statuses (see also [The Most Important Update Statuses \[Page 120\]](#)):

Update records that have not yet been processed are assigned the status *init* or *auto*. Both of these are normal statuses, in other words, you do not need to do anything here as long as the list is not too long, and the updates are not processed.

init indicates that an update record is complete (header, function module calls and data elements exist), but has not yet been processed.

auto indicates that a record is complete and will be processed after the update has been started. Records that could not be processed when they were created usually have the status *auto*

Selecting and Displaying Updates

(update server not active, update deactivated). The updates are processed automatically when the update is reactivated.

run indicates that an update record is being processed. Run is only assigned to automatic update records and records that are processed via *Repeat*.

err identifies update records that caused an update to be terminated prematurely because an error occurred while the update request was being executed. The messages **err(ext. commit)** and **err(no retry)** can also be output here. See also [Analyzing and Correcting Update Errors \[Page 165\]](#)

ok means that no errors occurred while the update request was being executed (if `rdisp/vb_delete_after_execution` is set to 2, in other words, automatic deletion after execution of update is deactivated)

V1 means that V1 modules of the update request were executed without any errors occurring; this does not, however, refer to the entire update request (that is, there are still V2 updates and/or collective runs to be processed).

V2 means that V2 modules of the update request were executed without any errors occurring; this does not, however, refer to the entire update request (that is, there are still collective runs to be processed).

del means that the update request is being deleted.



If you do not want to use time intervals or users as selection options, you can also use [Update Program Administration \(Transaction sm14\) \[Page 155\]](#).

Displaying the Update Header

Use

The **update header** is used to manage the update records ([update requests \[Page 112\]](#)).

Prerequisites

You have called up the `Update Records: Main Menu` screen and have positioned the cursor on an update record.

Procedure

You can display the update header by choosing *Goto → Update header*.

The update header contains the following information.

Field	Description
Update key	Key under which an update request is stored as a data record in the database. The key writes appropriate messages to the system log if update processing is canceled. The express mail that informs you of the canceled update also contains this key. Compare the key in an express mail or in the system log with the value in this field to make sure that you have found the correct update record.
Client	Client in which the user is logged on.
User	User who triggered the update
Language	Language in which the user is logged on
Account	The <i>Account</i> field is currently not used
Report	ABAP report with which the update was created
TCode	Transaction code that called up the ABAP program
ENQ key	Key of the lock entries created by the transaction that created the update. The update task holds the locks until the V1 update has been successfully completed. The locks are canceled if an update error occurs. This is described in greater detail in the documentation on The R/3 Lock Concept [Page 76] .
Context	Language that must be supported by the update server for this update request. Update group specified for the update. The context is only relevant if multiplexing for tables is active (Oracle Parallel Server database system). You can display the update server(s) in your system by choosing <i>Goto → Server (Language column)</i> .
Info	Bit mask that contains further information on the update request. Default value = 1 (standard update request). 2 means that this request is from the batch input, and must not be restarted manually .

Displaying the Update Header

Ud.ret.cde	Update return code (for error analysis)
Status	Detailed description of status
Update server	Update server on which the update was carried out
Error text	Error message (if present) indicating that the update was terminated prematurely.

Displaying Update Modules

Prerequisites

You have called up the `Update Records: Main Menu` screen and have positioned the cursor on an update record ([update request \[Page 112\]](#)).

Procedure

You can display the update modules of the selected update record by choosing *Goto → Update modules* or by double-clicking the record.

A list of the update modules is then displayed. This contains:

- the module ID, that is the number of the module
- the module name
- the module type (V1 NORMAL, V2 NORMAL, V1 NICHT SUB.PST., ARFC and COLL.RUN)
- the status of the module (*init*, *run*, *err* etc.), see also [The Most Important Update Statuses \[Page 120\]](#).

From this screen, you can display more detailed status information (position the cursor on the required module + `Update status` pushbutton), refresh modules (`Refresh` pushbutton), and display the update header (hat icon) and update data (glasses icon).

Analyzing Canceled Updates

Use

There can be many reasons why an update is canceled (`err` status in the overview). The problem often exists for a very short time only; in this case, the update records can be [processed manually \[Page 144\]](#). If the problem is of a more serious nature, you have to analyze it further.

See also [Analyzing and Correcting Update Errors \[Page 165\]](#)

Procedure

In the update management transaction, choose *Update records* → *Debugging* to analyze a canceled update in the debugger. Further information can be found in the [Debugger \[Ext.\]](#) documentation.

You can also correct the errors in an update record via the debugging function. To activate and deactivate the update debugging function choose *Settings* → *Update debugging on/off*.

In both cases, the debugger stops at the start of the update module called up by the transaction.



Authorizations: in order to use the debugging function, you require the update authorization for the authorization object *System authorizations* (S_ADMI_FCD) as well as the ABAP authorization for debugging *ABAP Development Workbench* (S_DEVELOP).

Testing Canceled Updates

Use

Performing a test is recommended in the following cases:

- You suspect that a temporary problem has occurred, or you have corrected the update problem and now want to test the update to determine whether it functions correctly. If the test is completed successfully, the problem has been corrected.
- You want to attempt to retrigger the error caused by the update problem. If the update is canceled during the test, the system outputs the error message with which the update was canceled. Click the message to call up online help.



An error message that is displayed during the test is not necessarily the original message because the error message always depends on the current system conditions.

See also [Analyzing and Correcting Update Errors \[Page 165\]](#)

Procedure

In the `Update records` screen, position the cursor on the record you want to test and choose `Update records` → `Test` to test the canceled update record.



Do not test any updates that are waiting to be tested or are in the process of being tested (status = `init`, `auto` or `run`) because processing could be canceled by the test. The different statuses are described in the section entitled [The Most Important Update Statuses \[Page 120\]](#).

Result

`Test` processes the selected function module and then outputs a message (in the status bar) indicating that the update was successful or indicating an error if the update was canceled.

`Test` does not make any changes to the database, or any changes made are reversed automatically after the update has been completed or canceled.

If the test was successful, you can run the canceled updates again ([Processing Updates Manually \[Page 144\]](#)).

Processing Updates Manually

Processing Updates Manually

Use

The function for processing update records manually is used to reprocess records, which for some reason did not successfully complete one of the phases in the update process.

This function can be used in most cases, and means that you do not have to delete the update records and enter the data again.

Prerequisites

The update records that have to be reprocessed are set to `err` on account of an update problem that occurred earlier, or `init`.

Procedure

The procedure used will depend on the status of the record at the time the problem occurred (see also [The Most Important Update Statuses \[Page 120\]](#)).

Update Record Set to `init`

You can instruct the Update System to process all updates with the status `init` (all records that have not yet been updated) by choosing *Update records* → *Update*. (If the update is active, the records are processed automatically; in other words, you do not need to do anything here). This is described in greater detail in the section entitled [The Most Important Update Statuses \[Page 120\]](#).

Update Record Set to `err`

You can instruct the Update System to reprocess all updates with the status `err` (canceled prematurely) by choosing *Update records* → *Repeat update*. To make sure that this functions correctly, you can [test the canceled updates \[Page 143\]](#) beforehand.



Updates for records that contain an update module of the type `V1 NICHT SUB.PST` **cannot** be repeated.



Note that the repeat update for a V2 module cannot be carried out before the repeat updates for V1 modules. Processing a V1 module can also trigger processing of the associated V2 modules. A V2 component is no longer displayed after it has been successfully processed. (Detailed information can be found in the section entitled [The Update Process \[Page 115\]](#)).

Result

The update records are processed correctly and no longer appear in the overview.

Further Information

[Repeating Canceled Updates \[Page 166\]](#)

Resetting Update Statuses

Resetting Update Statuses

Use

From time to time, an update that was set to `err` is not completed successfully when it is reprocessed, in other words, the update is always active (status = `auto` or `run`) and is neither fully processed nor terminated prematurely. (This problem only occurs when updates are repeated manually or when an update server is restarted if updates that have not been processed (status = `auto`) are processed automatically. Further details can be found in the section entitled [The Most Important Update Statuses \[Page 120\]](#).)

Two causes are currently known for this problem: a database problem of the type “dirty-read/dirty cluster” (because the update does not hold any locks for the data), or the target update server is not active when the update is repeated.

Under normal circumstances, the problem is only a temporary one (in both cases). Using transaction `sm51` and `sm50`, check whether the update is not processed. If it is not processed, reset it, and try to process it again (see also [Processing Updates Manually \[Page 144\]](#)).

Procedure

To reset the status of an update record, choose *Update records* → *Reset* → *Update status*.

Result

The update records are reset to the status `init` and have to be [processed manually \[Page 144\]](#).

Deleting Update Records

Use

Update records with the status `err` can be deleted.

Prerequisites

Do not delete the canceled updates (status = `err`)

- until you have completed the update error analysis. Remember that when update servers are started, they always search for updates that are older than 30 days and delete these. Once you have completed your analysis, you can stop the updates from being deleted automatically by increasing the value of the system profile parameter `rdisp/vbdelete` (see also [Main System Profile Parameters for Updates \[Page 128\]](#)).
- unless you are sure that the data in the update can be restored, if the update originates from a batch input session, for example, and can be repeated there, or if you have retrieved data from the canceled update so that you can enter it again.

Procedure

Choose *Update records* → *Delete* → *All records* or *Single*.

Single refers to the update record at which the cursor is currently positioned.



Never delete updates that have not yet been processed (status = `init`, `auto` or `run`) as this can result in data that is to entered in the R/3 System being lost.



Deleting Updates in the Background

The function for deleting updates automatically after they have been successfully completed can be deactivated in order to prevent the system constantly deleting small numbers of update records (extremely high number of database accesses).

Deactivating this automatic delete function can, however, result in very long update tables in the database. To remove the updates, you have to run the report RSM13002 at least once a day (the option *DELETE* must to be set to X). Only updates that have been completed successfully are deleted with the report RSM13002.

Further information can be found in the description of the parameter `rdisp/vb_delete_after_execution` in the section entitled [Main System Profile Parameters for Updates \[Page 128\]](#) or in transaction rz11 in the R/3 System.

Result

The update record(s) is/are deleted, the data is lost and has to be reentered. All of the locks on database objects held for the update are also deleted (see also [The R/3 Lock Concept \[Page 76\]](#)).

Reorganizing Update Records

Reorganizing Update Records

Use

You can use this function to instruct the update server to search for incomplete update records and delete these. Update records can be incomplete - if a transaction registers one or more update modules - as a result of a screen change in which the transaction that creates the update generates a database commit; the transaction, however, is terminated prematurely and a rollback is triggered.

Incomplete update records are of no significance, but take up space in the database.

If the parameter [rdisp/vbreorg \[Page 128\]](#) is set to 1, the update records are reorganized each time the update server is started.

Reorganization is, therefore, active (value = 1) in the standard system.

This function allows you to reorganize the update records manually.

Procedure

Choose *Update records* → *Reorganize*.

Result

Incomplete update records are deleted and do not occupy valuable space in the database.

Configuring Update Groups

Use

Automatic load distribution takes place in an R/3 System even with asynchronous updates. Load distribution ([update dispatching with load balancing \[Page 125\]](#)) usually takes place across all of the available update work processes.

This global dispatching is, however, not always desired.

Update work processes can, therefore, be combined by R/3 application servers to form groups within which load distribution (dispatching) takes place. **Dispatching does not take place outside these groups.**



You can activate dispatching by setting the profile parameter `rdisp/vb_dispatching` to 1 (see also [Main System Profile Parameters for Updates \[Page 128\]](#)). If this parameter is set to 0, all of the update requests are sent to the server specified in the profile parameter `rdisp/vbname`.

The procedure for creating the groups is described below.



As an alternative to the procedure described below, you can also configure server groups with the much more user-friendly [Update Program Administration \(Transaction sm14\) \[Page 155\]](#).

Procedure

1. To create groups, call up transaction `se16` (table maintenance) and make the following entries in the `SERVERGRP` and `DBINSTANCE` fields of the table `ASGRP`:

SERVERGRP	DBINSTANCE
GROUP1	<SID>
GROUP2	<SID>
...	...

<SID> represents the SAP system name. "GROUP1" and "GROUP2" are freely selectable unique names.



Let us assume that you want to create one update group for financial accounting documents and one for production planning. With the SAP system name `<SID> = C11`, the table `ASGRP` could contain the following entries:

SERVERGRP	DBINSTANCE
FI_GROUP	C11
PP_GROUP	C11

Configuring Update Groups

...	...
-----	-----

2. The following entries are made in the SERVERNAME and SERVERGRP fields of the table APSRV:

SERVERNAME	SERVERGRP
<app_server_name>	GROUP1
<app_server_name>	GROUP1
...	...
<app_server_name>	GROUP2
<app_server_name>	GROUP2
<app_server_name>	GROUP2
...	...

The entries should reflect the configuration of your system. The name of the application server is structured as follows: <host>_<SID>_<Num>.



Make sure that the server name is written correctly (call up transaction **sm51**, which displays the names of the servers).



The table APSRV could then be as follows:

SERVERNAME	SERVERGRP
app1_C11_00	FI_GROUP
app2_C11_00	PP_GROUP
app3_C11_00	PP_GROUP



Servers that are not assigned to any group in the table APSRV, are assigned implicitly to the group "DEFAULT". If this group does not contain an update server, no updates can be performed by these servers (the message "No update server active" is output when you save your data).

3. Stop and then restart all R/3 instances.



Modified configuration: if the application server assignments are changed, or if a new application server is defined in the system, you only have to carry out steps 2 and 3.

Diagnosis:

Start the test updates on the application servers of the individual groups: in transaction **sm12**, *Extras* → *Diagnosis in update*, you can check to see whether the groups were created successfully.

Detail on monitoring updates can be found in the section entitled [Monitoring Updates \[Page 161\]](#); details on transaction **sm12** can be found in the section entitled [Managing Lock Entries \[Page 98\]](#).

Displaying and Resetting Update Statistics

Displaying and Resetting Update Statistics

In the update management initial screen (Update Records: Main Menu), choose *Goto* → *Statistics* to display the update activities on the server you are currently logged onto. The statistics contain all of the data collected as of the last reset or server restart.

To reset the statistics, choose *Update records* → *Reset* → *Statistics*.

You can now specify whether the statistics are to be reset only on the application server you are currently logged onto (*local*), or on all servers (*global*).

The statistics are divided up into the following sections:

Update requests

Update requests			
generated	1	executed (V1)	15
started (V2)	2	executed (V2)	2
canceled	0	deleted	16
Update is active			

The client is displayed on the left and the server on the right. The [update request \[Page 112\]](#) is sent from a dialog or batch work process (client) on the application server you are working on (see status bar) to the update work process (server), which may be running on a different application server.

The **1st line** specifies how many update requests were generated by the client (in other words, sent to the server), and how many were processed by the update work process (executed (V1)). This number is higher because you are on the application server on which the update server is running - this also processes update requests from other application servers.



If you call up the statistics on a server on which no update work process is running, 0 always appears alongside executed (V1), started (V2) and executed (V2).

As described in the section entitled [The Update Process \[Page 115\]](#), the update work process passes the V2 update modules onto a V2 work process – in other words, the **2nd line** contains the update work process of the clients and the V2 work process of the servers. Furthermore, both V2 updates were carried out.

In the **3rd line** you can see how many update requests were canceled by the client, and how many were deleted by the server.

The **4th line** indicates whether the update is active (see also [Deactivating and Reactivating Updates \[Page 164\]](#)).

Database Accesses

DB I/O	bytes written	bytes read
--------	---------------	------------

Displaying and Resetting Update Statistics

in total	456	119949
min	28	28
avg	152.000000	325.065041
Max	356	1172

This section logs the database accesses made as a result of the update: total number of bytes written and read, the smallest and largest packet, as well as the average size. This is, therefore, the data that is written to the update tables VBHDR, VBMOD, VBDATA and VBERROR.

Times

Times	Execution (V1)	Execution (V2)	Read	Write
Number	17	2	375	3
total (sec)	9.734802	6.046222	2.349286	0.113074
min (msec)	46.373000	3006.741000	0.125000	32.659000
avg (msec)	572.635412	3023.111000	6.264763	37.691333
max (msec)	3613.746000	3039.481000	355.894000	44.082000
kb/sec			50.306175	3.938240

This section contains information on the time required to execute the update requests. **Column 1:** number of V1 update requests executed, the total time required for these in seconds, as well as the shortest, average and longest time required in milliseconds.

Column 2: the same information as column 1 but, in this case, for V2 updates.

Column 3: number of read operations (updates tables in the database), the total time required for these in seconds, as well as the shortest, average and longest time required for a read operation in milliseconds.

Column 4: the same information as column 3 but, in this case, for write operations

Times	Update	Commit times	Delete
Number	5	42	54
total (sec)	0.183918	1.248347	0.283877
min (msec)	3.770000	0.217000	2.474000
avg (msec)	36.783600	29.722548	5.256981
max (msec)	122.488000	188.723000	66.108000

This section contains the time intervals required for the different steps (minimum, maximum, and average values). Update table updates (column 1), commits and delete operations in the update work processes.

Displaying Servers

Displaying Servers

Use

This display (initial screen: `Dispatching info`) contains the following information:

- the update groups created (see also. [Configuring Update Groups \[Page 149\]](#))
- the active update servers and associated work processes (`upd1` for V1 updates, `upd2` for V2 updates), group affiliation, languages supported, and update requests that have just been allocated
- the other application servers that are active in the R/3 System, the proposed work process type (D=Dialog, B=Batch(background), S=Spool, E=Enqueue) and group affiliation

Procedure

In the update management initial screen, choose *Goto* → *Server*.



You can also display the servers in [Update Program Administration \(Transaction sm14\) \[Page 155\]](#) (see also [Server tab page Server \[Page 158\]](#)).

Update Program Administration (Transaction sm14)

Use

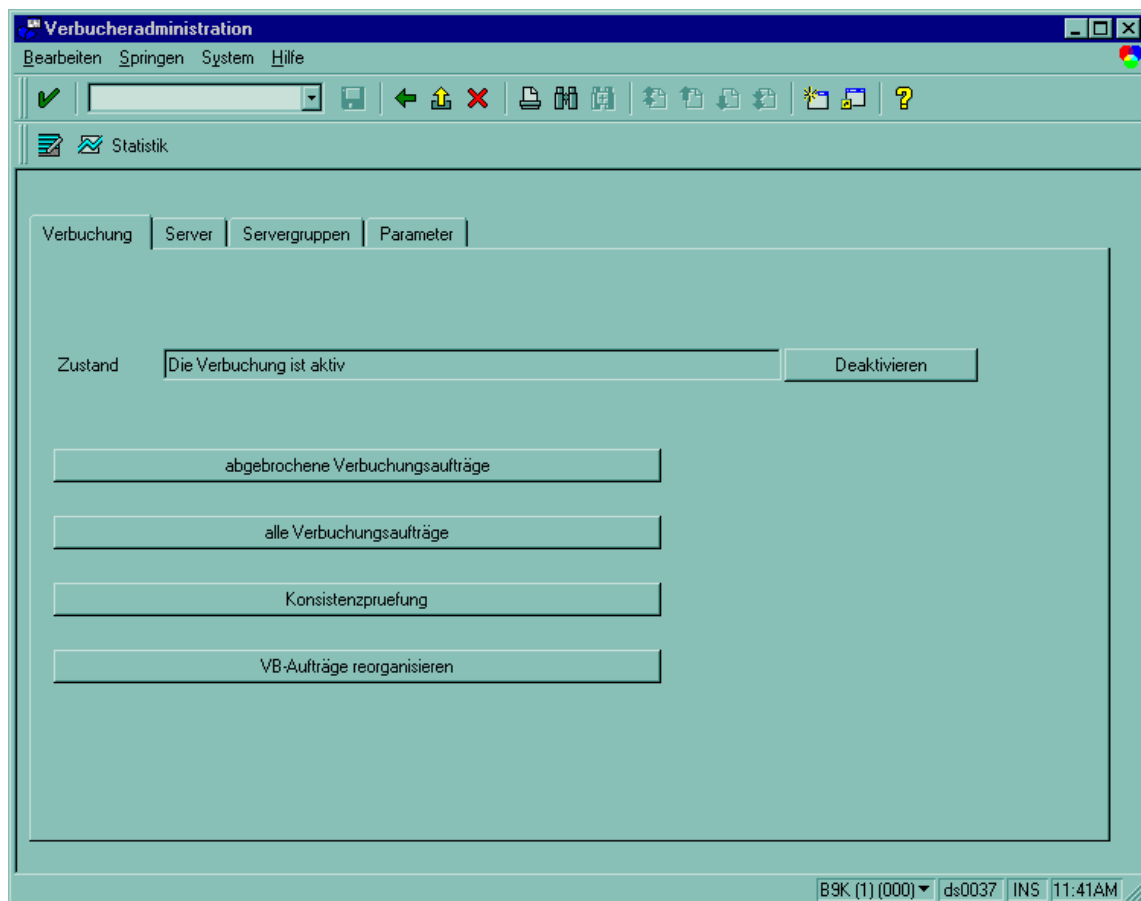
Compared with transaction **sm13**, **sm14** provides you with a different view of an update ([Update Management \[Page 135\]](#)) and contains additional functions for administering updates (for example, creating servers and server groups, checking parameters relevant for updates).

Integration

You can access all of the functions supported by **sm14** via the appropriate menu in the initial screen of transaction **sm13** or via other transactions. Transaction **sm14** merely groups together related functions so that you can use them more easily.

Features

The initial screen is shown below (update is active):



The initial screen contains four tab pages:

- [Update tab page \[Page 157\]](#) (shown here)
- [Server tab page \[Page 158\]](#)

Update Program Administration (Transaction sm14)

- [Server groups tab page \[Page 159\]](#)
- [Parameter tab page \[Page 160\]](#)

Activities

You can call up Update Program Administration by entering transaction code **sm14** (or **/nsm14** if you want to call it up directly from a different transaction).

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Documentation not available for Release 4.6C

Setting Up Locals Executables on UNIX R/3 Instances

Use

If you have a distributed homogeneous R/3 System, then you can choose how the distributed instances are to access the executable files (programs) of the R/3 System. Here, homogeneous means that the host on which the various R/3 instances run, all have the same UNIX operating system.

In a heterogeneous R/3 System made up of instances running on different UNIX operating systems, or running on UNIX and NT, the aforementioned does apply universally since the since the various platforms cannot use their executables in common.

Implementation Considerations

In a homogeneous UNIX-R/3 System, you can choose between a standard installation (see [Structure of Executables after the Standard Installation \[Page 170\]](#)) and an [installation of local executables with automatic adjustment \[Page 175\]](#).

For more information on the differences, and advantages and disadvantages of both choices see [The Differences Between Central and Local Executables \[Page 171\]](#).

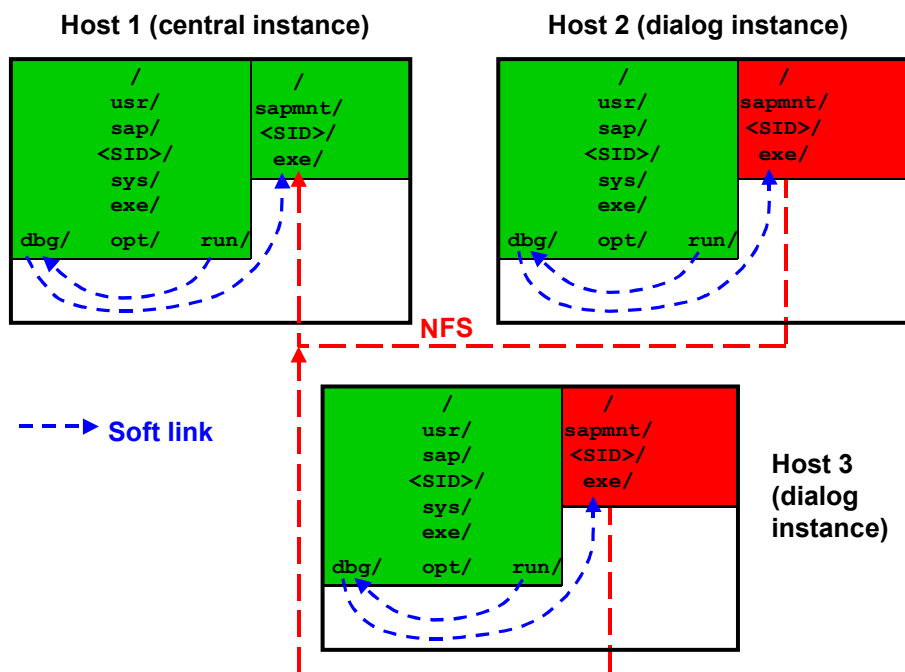
Structure of Executables after Standard Installation

Structure of Executables after Standard Installation

In a standard R/3 System, executables (program files) are stored in a single executables directory. The name of this directory is operating-system specific. The following section explains the UNIX configuration.

After a standard installation with several instances in a homogenous R/3 System, the executables are located in a directory on a central server (host or host system). Normally, it is the host on which the central instance of the R/3 System was created. Normally, additional dialog instances are also installed on other hosts. These distributed R/3 instances use the central executables directory together over your network using NFS. See the following graphic.

File Structure after the Standard Installation



In UNIX systems, the name of the default directory for the executables used by R/3 is `/usr/sap/<SID>/SYS/exe/run`. The directory contains a soft link to either the `dbg` directory or the `opt` directory (`dbg` is the default here).

The `dbg` directory contains executables that can be debugged (or a soft link to a directory that contains such files). The `opt` directory contains optimized executables without troubleshooting information. The `opt` directory can also contain a soft link to a directory containing optimized executables.

Executables, Difference between Central and Local

After a standard installation, you can do the following:

1. **Putting the central storage for the executables on a central server.** The executables can be used commonly through distributed instances over the network (NFS).

Advantages	Disadvantages
<p>You need to provide storage for executables only once, on a central system in your network.</p> <p>If the executables are changed, all the instances are automatically provided with the correct executables after a restart.</p>	<p>Startup of instances is slow because executables have to be loaded across the network rather than from local disk storage.</p> <p>Central storage can result in workload problems that lower performance.</p> <p>If the hosts of distributed R/3 instances run out of memory, then they must page active executables over the network. This can result in heavy workloads on the network and on the central system where the executables are stored.</p>

See also: [Structure of the Executables after the Standard Installation \[Page 170\]](#)

1. **Putting local storage of executables in distributed R/3 instances while using automatic adjustment.**

If the executables were updated in the central directory, each R/3 instance updates its executables when the system starts up.

With this procedure, you can specify for each instance, if it takes part in the automatic executable adjustment, or if it works with its local executables.

See also: [Functions of the Automatic Adjustment \[Page 172\]](#)

SAP recommends storing executables locally if the host systems in your network have adequate disk storage. To store R/3 executables locally, a host system must have several hundred MB of free disk storage (depending on the R/3 Release).



Not all of your R/3 instances must use local executables. You can create a “mixed” configuration in which the executables are stored locally only on certain host systems. The host systems with less memory space then access the central executables.



If the prerequisites for the automatic executable adjustment are only partially met, or not at all, the executables are not automatically kept at the same release level on the various instances. If you upgrade your system, or import Hot Packages, you risk forgetting to update your executables on the individual dialog instances.

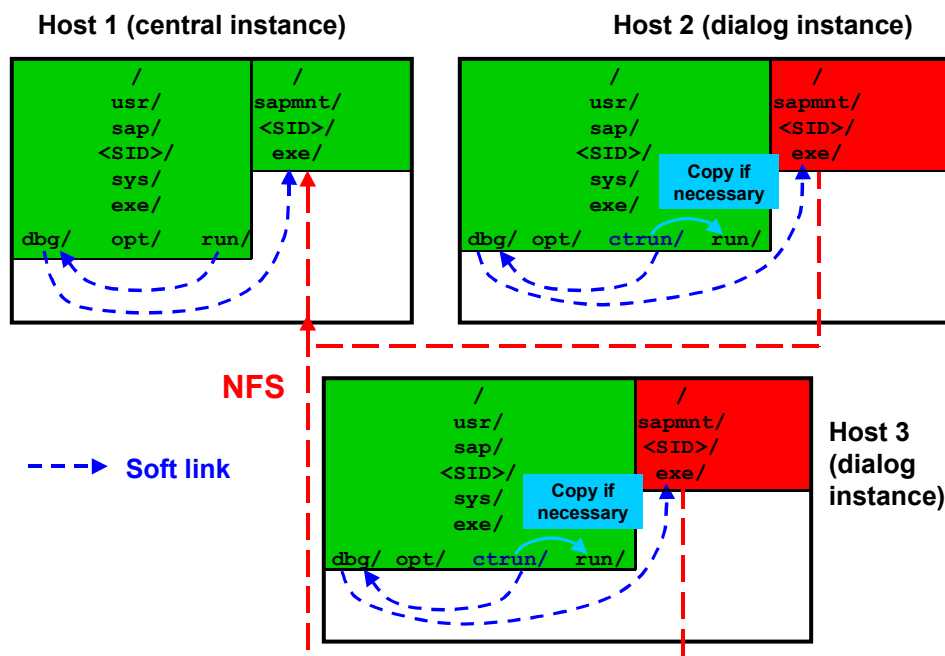
Functions of the Automatic Adjustment

Use

When you restart the SAP instance using command `startsap`, the program `sapstart` is automatically called. This program automatically calls the [program `sapcpe` \[Page 177\]](#). The program `sapcpe` ensures that the automatic adjustment of the executables is made by copying `/usr/sap/<SAPSID>/exe/ctrun` from the central directory to the local `/usr/sap/<SAPSID>/exe/run` directory.

At all subsequent startups, `sapcpe` checks that the local executables are still current and copies any new or altered executables from the central `/usr/sap/<SID>/SYS/exe/ctrun` directory.

File Structure for Using Local Executables



Prerequisites

If you start an R/3 instance and the following 3 conditions are met, `sapcpe` starts automatically:

- The directories `/usr/sap/<SID>/SYS/exe/run` and `/usr/sap/<SID>/SYS/exe/ctrun` exist.
`/usr/sap/<SID>/SYS/exe/run` is located on the host system of the R/3 instance and is for locally-stored executables.
`/usr/sap/<SID>/SYS/exe/ctrun` accesses (normally using NFS mount) the central directory of the executables.
- The directories `/usr/sap/<SID>/SYS/exe/run` and `/usr/sap/<SID>/SYS/exe/ctrun` are in the same file system.

Functions of the Automatic Adjustment

- The directory `/usr/sap/<SID>/SYS/exe/run` must be local (physically attached to the host system on which the R/3 instance is started), and must have enough storage space for the R/3 executables.

How you set up your system is described in [Installing Local Executables with an Automatic Adjustment \[Page 175\]](#).

Features

Log File

If **sapcpe** is called from **sapstart**, it writes all actions to the log file in the local directory `/usr/sap/<SAPSID>/<INSTANZ-NAME>/work/sapcpe.log`.

Compressed Executables Files

sapcpe can work with compressed executable files (`<file name>.Z`) in the central directory.

If you have a large number of distributed R/3 instances (more than 5), SAP recommends storing executables in compressed form in the central directory.

The benefit: Compressed files reduce the network load caused by copying executables by approximately 50%.

sapcpe detects changes in compressed files using the time stamp. If the compressed file is newer than the local copy, **sapcpe** copies the compressed file from the central directory to the local directory and decompresses it.

Access Authorization and Ownership

There are no special access authorizations required for **sapcpe**. Therefore, take the following security precautions:



Do not set the **set user ID bit** (UNIX systems) for **sapcpe** authorization.

Since **sapcpe** is executed as user `<SID>adm`, certain programs cannot be updated. The program also cannot update programs which are active when **sapcpe** tries to update a local system. If **sapcpe** encounters ownership or authorization problems, it records the problems in its log file. You must then update the relevant executables manually, or correct the problem and restart the instance.

Normally, only the SAP operating system monitor **saposcol** is affected by this. Program **saposcol** is a part of the R/3 performance monitor and therefore, is the only SAP program that must run as `root`.

saposcol is started when the R/3 System is started. As soon as a **saposcol** error message is logged in the **sapcpe** log file displaying that **saposcol** must be updated, then you must stop the program and copy the executable manually from the central executable directory to the local executable directory.



Do not update **saposcol** by running the program from the central executables directory. The program may terminate abnormally if paging over the network is so slow that it produces time-outs.

Functions of the Automatic Adjustment**Activities**

Install the local executables using the automatic adjustment as described in [Installing Local Executables using Automatic Adjustment \[Page 175\]](#).

Installing Local Executables with Automatic Adjustment

To have the R/3 System store executables locally for an R/3 instance and automatically keep them up to date with the central copies, perform the following procedure.

Prerequisites

Required Disk Space

Ensure that each host server that is to use local SAP executables has several hundred megabytes (depending on the R/3 Release) of free disk space for the executables.

Procedure

Carry out the following actions as the user <sid>adm.

1. Change the name of the standard directory containing the variables from
`/usr/sap/<SID>/SYS/exe/run` to `/usr/sap/<SID>/SYS/exe/ctrun`.



in an R/3 System with the system ID **C11**:

```
mv /usr/sap/C11/SYS/exe/run sr/sap/C11/SYS/exe/ctrun
```

2. Ensure that the central SAP executables directory
`/sapmnt/<SID>/exe` is accessible via `/usr/sap/<SID>/SYS/exe/ctrun`.

A standard installation usually contains a symbolic link or **softlink** from
`/usr/sap/<SID>/SYS/exe/ctrun` to the local directory
`/usr/sap/<SID>/SYS/exe/dbg`. The directory `/usr/sap/<SID>/SYS/exe/dbg`, in
turn, has a symbolic link to the central directory `/sapmnt/<SID>/exe` (see graphic
under [Functions of the Automatic Adjustment \[Page 172\]](#)).



Command for displaying soft link assignments:

```
ls -l /usr/sap/C11/SYS/exe
```

3. On each host system that is to use local executables, create the local directory
`/usr/sap/<SID>/SYS/exe/run`.



```
mkdir /usr/sap/C11/SYS/exe/run
```

The directory must be local to each host system. That is, it must be located on a disk that is physically connected to the host system.

Alternative: create a local executables directory using any name that you wish and define a softlink from `/usr/sap/C11/SYS/exe/run` to this directory.



```
mkdir <executables directory>
```

```
ln -s <executables directory> /usr/sap/C11/SYS/exe/run
```

Installing Local Executables with Automatic Adjustment

4. Copy the `sapstart` program to the local executables directory.



```
cp /usr/sap/C11/SYS/exe/ctrun/sapstart  
/usr/sap/C11/SYS/exe/run/sapstart
```

See also: [sapcpe Program \[Page 177\]](#).



Note that the specified path names and commands are intended for the UNIX environment. If you are using a different platform, you must adapt these specifications to your operating system. The objective here is to build the constellation shown in the graphic under [Functions of the Automatic Adjustment \[Page 172\]](#).

Program sapcpe

Use

Program **sapcpe** ensures [automatic adjustment \[Page 172\]](#) of locally installed executables, if changes have been made to the executables on the central instance.

Prerequisites

See [Installing Local Executables using Automatic Adjustment \[Page 175\]](#).

Features

Program **sapcpe** does the following:

- It performs the initial copy of executables from the central directory to the local directory if you set up a system for local executables (see [Installing Local Executables using Automatic Adjustment \[Page 175\]](#)).
Only the most often used executables are copied. For executables that are used less often, **sapcpe** sets up soft links to the executables in the central directory (see graphic in [Features of the Automatic Adjustment \[Page 172\]](#)). Which executables are copied, and which executables have a soft link is determined in special files. This is covered in a later section.
- It checks that the local executables are up to date at each startup of an R/3 instance that uses local executables. **sapcpe** checks the local executables against the central directory. It copies new executables whose date or size has changed.

This **sapcpe** function makes it unnecessary to distribute executables manually after installing an R/3 Release or upgrade. To update your distributed R/3 Systems, you only need to restart the R/3 instances that are set up for the automatic adjustment.

If **sapcpe** finds any of the following list files in `/usr/sap/<SID>/SYS/exe/ctrun`, it copies only the executables listed in these files to the local `/usr/sap/<SAPSID>/SYS/exe/run` directory:

`instance.lst, instancedb.lst, tools.lst, inhouse.lst, frontend.lst`

If these files are missing, **sapcpe** copies all of the files listed in its own internal file table. This list contains all the executables necessary for an R/3 instance to operate normally. All the executables in this list are copied to the local directory. For example: `gwrld`, `gwwp`, `gwhost`, `sapexec`, `sapmscsa`, `disp+work`, `rsyn.bin`, `saprouterr` etc.

You can override the internal table using the entry in the file `sapcpeft`. For more information on these files, see [Configuring sapcpe \[Page 179\]](#).

By default, **sapcpe** uses soft links to make other, less frequently-used executables available in `/usr/sap/<SID>/SYS/exe/ctrun`. This procedure ensures that these executables, which generally are used only for testing, do not fill local disk space.

Note that the initial startup after an update can delay the start of the instance for several minutes due to the executables being copied.

Activities

[Configuring sapcpe \[Page 179\]](#)

Configuring sapcpe

Generally, you do not need to customize or configure sapcpe. Should you need to change the way the program works, you can do so with command line options, or by creating or editing control files. If you have called up **sapcpe** from **sapstart**, you cannot change the transfer parameters to **sapcpe**. In this case you can only modify the files used by **sapcpe**. If you call up **sapcpe** directly, you can control the program by entering transfer parameters. Calling up **sapcpe help** gives you a list of the parameters that you can set.

You can define which files should be copied, or should have links, from the central directory to the local directory by calling up **sapcpe** directly and setting two control parameters.

The first parameter, **all**, has the highest priority and ensures that all files in the central directory are processed.

The second parameter is **list:<filename>**. If this parameter has been set, only all files from the list file <filename> will be processed.

If neither of these parameters has been set, the program searches for list files in the central directory.

Known list files:

- **instance.lst**
List of database-independent executables. These executables are valid for all database systems with which the R/3 System runs.
- **instancedb.lst**
List of database-dependent executables. These executables are valid only with a particular database system.
- **tools.lst**
all files that do not belong in either instancedb.lst or instance.lst.
- **inhouse.lst**
all local files which should not be replaced by **sapcpe** during a copy run.
- **frontend.lst**

If there is at least one list file in the central directory, then the files entered in this (these) list file(s) are created. If there is no list file in the central directory, then the files in the internal program list are created in the local directory.

Either a soft link or a copy may be created, or a simple check of the availability of the file may be carried out. This depends on

- the internal table or
- the **sapcpeft** file or
- the file transferred with the **ftable** control parameter.

The **ftable** control parameter has the highest priority, and the availability of the **sapcpeft** file has the second-highest priority. Files of the **ftable** type (including the **sapcpeft** file) have the following structure:

Configuring sapcpe

Program names | attributes

disp+work | local_copy

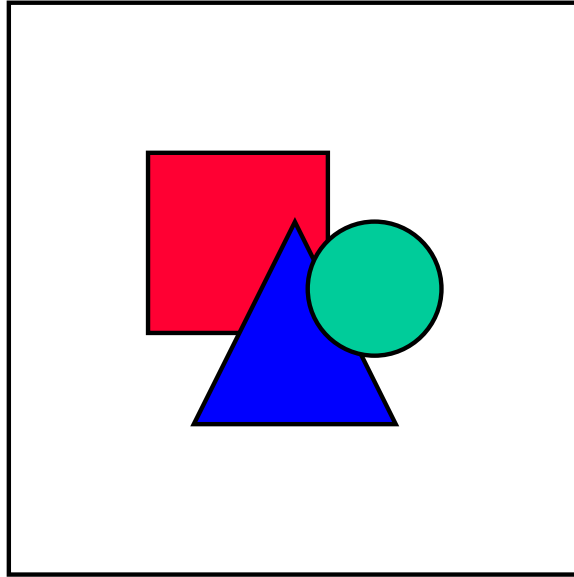
Possible attributes: local copy, softlink, check_exist

Managing Batch Input Sessions

Batch input sessions enter data non-interactively into an R/3 System. Batch input is typically used to transfer data from non-R/3 Systems to R/3 Systems or to transfer data between R/3 Systems.

This section describes how to manage batch input sessions.

To reach the main menu of the batch input system, select *System* → *Services* → *Batch input*. Alternatively, enter transaction SM35.



For information on the *Recording* function in the batch input manager (transaction SHDB), please see the online guide [Basis Programming Interfaces \[Ext.\]](#). You can use the recording function to create a batch input session without having to write or run a program to generate a session.

Overview and Concepts

[Process Overview: Batch Input \[Page 183\]](#)

[Batch Input: Concepts \[Page 186\]](#)

Working with Batch Input Sessions

Managing Batch Input Sessions

[Processing Sessions Automatically \[Page 188\]](#)

[Selecting Sessions \[Page 189\]](#)

[Running Sessions \[Page 192\]](#)

[Correcting a Session \[Page 196\]](#)

[Deleting Sessions \[Page 200\]](#)

[Locking and Unlocking Sessions \[Page 201\]](#)

[Releasing and Restarting an Interrupted Session \[Page 202\]](#)

Information and Analysis

[Displaying Queue Management Information \[Page 204\]](#)

[Displaying Session Logs \[Page 205\]](#)

[Reorganizing the Log File \[Page 206\]](#)

[Analyzing Sessions \[Page 194\]](#)

Process Overview: Batch Input

Purpose

Batch input is one of the primary ways in which data can be transferred into the R/3 System. Batch input is used for bulk data transfers and not for near real-time data transfers.

Typical uses of batch input include the one-time import of data from a legacy system into a newly installed R/3 System. Another typical use is for periodic (hourly, daily...) transfers of data from external systems or legacy systems that are still in use into R/3, where all enterprise data is consolidated.

For the system administrator, batch input usually requires minimal attention, only a periodic check to make sure all batch input sessions have been processed successfully. In the event of an error in a session, then the responsibility for triggering the error analysis and correction of the problem usually lies with the system administrator. Detailed analysis of the problem will require the help of the department or specialist responsible for the data to be entered. Incorrect entries in an R/3 financials transaction in a session, for example, have to be evaluated and corrected by the book-keeping department.

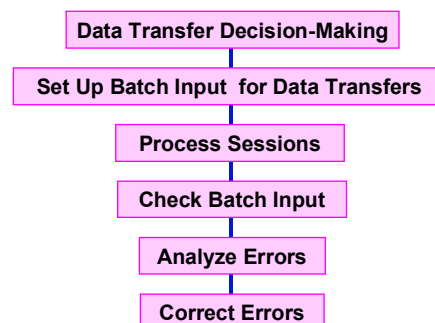
The process description here explains the typical workflow in batch-input operation from the point of view of the system administrator.

Prerequisites

The R/3 applications deliver many programs for batch input that are ready to use. Sometimes, however, a customer must write his or her own batch input program in order to convert data from a legacy system or from a proprietary format into an R/3 data format.

Batch input operation occurs only if a customer explicitly sets it up. The customer must activate a program that prepares a batch input session and that delivers it to the R/3 System.

Process Flow



- **Data transfer decision-making:** The batch input process begins with a decision to transfer data from an external source into R/3. The external source may be a legacy system that is being replaced. In this case, a one-time bulk data transfer is foreseen. For example, customer data from a customer's old financials system may be transferred with batch input to a new R/3 System.

Process Overview: Batch Input

Alternatively, the external source may be an external system that is to remain in use, another R/3 System, or an R/2 System. In this case, a regularly-recurring bulk data transfer is foreseen. For example, data from a free-standing engineering system may be taken over into R/3 by means of a daily batch-input run.

- **Setting up batch-input for data transfers:** If R/3-standard one-time or regular data transfers are required, then these may be set up by means of customizing settings in the R/3 Customizing System in SAP ASAP. Example: Batch input data transfers can be activated for legacy data in the Materials Management application (see 'Data Transfer').

Custom batch input procedures must be set up by hand. That is, the system administrator must schedule the data conversion program that creates the batch input session. How frequently data is made available from the external system, how frequently the conversion program should run, and whether the conversion program runs in R/3 (ABAP program) or in a host system (external program) must all be determined by the system administrator and the batch input programmer. See also the online guide [Basis Programming Interfaces \[Ext.\]](#).

- **Processing batch input sessions:** The actual transfer of data into R/3 takes place when a batch input session is processed.

For the system administrator, processing of batch input sessions requires little attention. Usually, system administrators automate the starting of batch input sessions (see [Processing Sessions Automatically \[Page 188\]](#)). Administrators can also start batch input sessions explicitly from transaction SM35, if necessary.

- **Checking batch input sessions:** The only routine activity for a system administrator is to check daily or more frequently in transaction SM35 that all batch input sessions have been completed successfully. The schedule for checking sessions depends upon the schedule for running batch input sessions. The R/3 System provides easy-to-use batch input management tools for doing this check (see [Selecting Sessions \[Page 189\]](#)).
- **Analyzing errors:** If one or more transactions in a session ended in errors, then the system administrator must analyze the problem. Usually, the system administrator will need the assistance of the affected data entry specialist or department for this analysis. The programmer who wrote the data conversion program may also need to be involved, if the problem was caused by incorrect data conversion or incorrect generation of the batch input session.

The batch input system offers detailed logging and powerful analysis tools to help you find out the cause of a problem (see [Analyzing Sessions \[Page 194\]](#) and [Displaying Session Logs \[Page 205\]](#)).

Most problems fall into one of two categories:

- Required data is missing from the batch-input session or invalid data has been included in the session. Possible external causes of this type of problem include errors in the data conversion program or the presence of unexpected types of data or incorrect data in the legacy database. Causes for this type of problem within R/3 include incorrect or incomplete customizing in an application. For example, a legacy data type may not have been foreseen in the check table entries made in application customizing.
 - Technical/programming problems. A batch input session enters data by running R/3 transactions non-interactively. A typical technical or programming problem is therefore incorrect identification of one of the data fields in a transaction. Or the conversion program may not fill a required data field or may have provided invalid values.
- **Correcting errors:**

Process Overview: Batch Input

The batch input system processes all correct transactions in a session. It also guarantees that successfully completed transactions in a session cannot be run again. To correct transactions with errors, the system administrator or the responsible department can interactively correct and reprocess the transactions. See [Correcting a Session \[Page 196\]](#).

Batch Input, Fast Input, and the Data Transfer Workbench

Batch input is one of several techniques that are in use for transferring data into the R/3 System. Different R/3 applications and SAP components may use batch input, fast input, or data transfer by way of calls to a BAPI interface.

All of these data transfer techniques are brought together in the data transfer workbench. In the workbench (transaction BMV0), you can find all batch input, fast input, and BAPI data transfer programs. And you can branch to transaction SM35 to run batch input sessions from the workbench.

You can find information on the technical implementation of batch input and fast input and on the programming of these techniques in the online guide [Basis Programming Interfaces \[Ext.\]](#). Here you will also find information on how these two techniques for transferring data differ.

BAPI-based data transfer interfaces are individually documented in the Business Object Repository.

Result

Batch input is a key technology for integrating R/3 into heterogenous ERP environments. Batch input provides the capabilities necessary to ensure a smooth and efficient flow of information into R/3 from external sources.

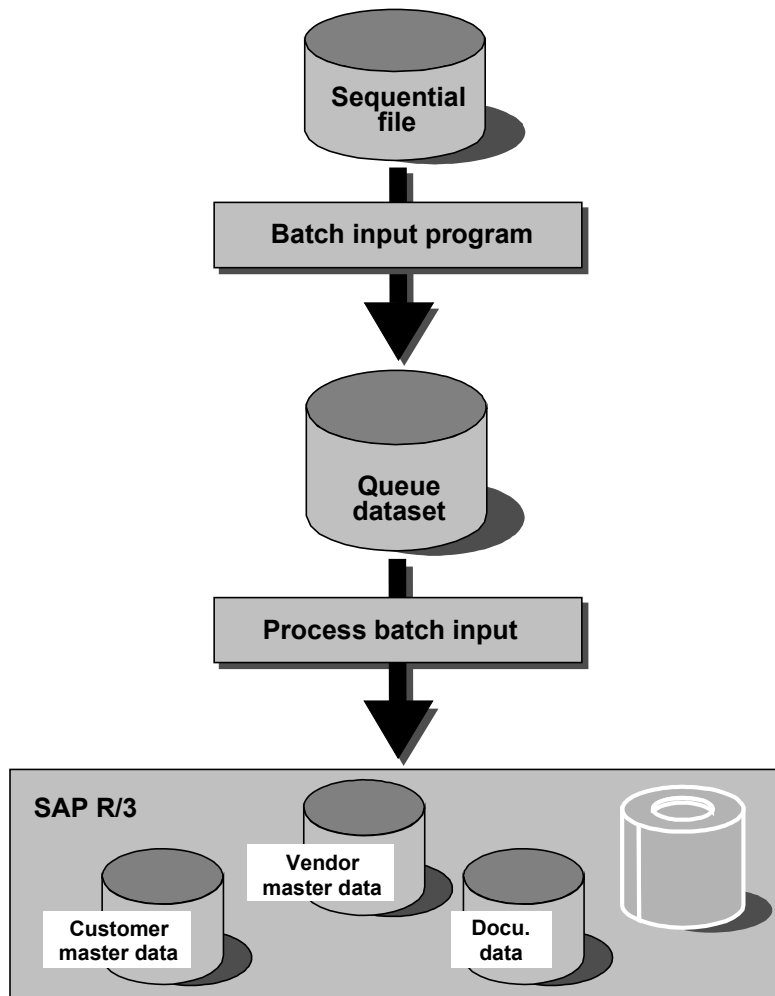
Batch Input: Concepts

Processing Sessions

A batch input session is a set of one or more calls to transactions along with the data to be processed by the transactions. The system normally executes the transactions in a session non-interactively, allowing rapid entry of bulk data into an R/3 System.

A session records transactions and data in a special format that can be interpreted by the R/3 System. When the System reads a session, it uses the data in the session to simulate on-line entry of transactions and data. The System can call transactions and enter data using most of the facilities that are available to interactive users.

For example, the data that a session enters into transaction screens is subject to the same consistency checking as in normal interactive operation. Further, batch input sessions are subject to the user-based authorization checking that is performed by the system.



Generating Sessions

To transfer data with batch-input, the system that is sending the data uses a data transfer interface provided by an R/3 application program in the receiving system. The interface program in the application then produces a batch input session.

The interface program in an application is an ABAP/4 program that sets up the transaction calls and data that make up a session. If the batch input session contains data from an external source, the program also reformats the data to meet the requirements of the input fields in which the data is to be entered. Usually, such programs are provided by the R/3 applications. For more information, please see the documentation of the R/3 applications and the online guide [Basis Programming Interfaces \[Ext.\]](#).

Authorizations

When a session is generated, a client and user are associated with it. If the session is run in background-processing mode, the system uses this user for authorization checking as the session runs. This authorization testing applies whether you sent the job for batch execution or the session was started by a background job.

Sessions that you process in one of the interactive modes are run with your authorizations. The interactive modes are described later in this section.

Processing Sessions Automatically

Processing Sessions Automatically

Use

In most cases, batch input sessions can be processed automatically. It is not necessary for a session to wait until a system administrator explicitly starts the processing of the session.

This section explains how to have sessions started automatically soon after the session has been generated in an R/3 System.

Prerequisites

The ABAP program RSBDCSUB must be scheduled as a periodic job in the R/3 background processing system. RSBDCSUB checks for and starts any batch input sessions that have not yet been run. It schedules such sessions for immediate execution in the background processing system.

Procedure

Schedule RSBDCSUB to run periodically in one or more background jobs.

If you have regularly scheduled batch input runs, you can schedule separate jobs for each of the scheduled data transfers. The start time for the RSBDCSUB job can be set according to the batch input schedule. And you can use a variant to restrict RSBDCSUB only to the batch input sessions that you expect.

With RSBDCSUB, you can use all of the selection criteria offered on the batch input main menu to select sessions to run:

- session name
- date and time of generation
- status: ready to run or held in the queue because of errors

Result

Batch input sessions are started automatically rather than by hand. The RSBDCSUB program can be set up to start all sessions that arrive in an R/3 System, or it can be fine-tuned to start only batch input sessions that you expect.

Selecting Sessions and Reading the Session Display

Use

The heart of the tool for managing batch input sessions is the session list. The management tool offers lists by status, and you can select lists according to other criteria as well.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

On the main menu, you can select sessions using any or all of the following criteria:

- session name
- date on which the session was generated (entered into the session queue)
- user who generated a session
- session status

In the session overview, you can start, analyze, or delete sessions in the queue. You can also display statistics on the transactions in a session and view the session log. All sessions generate a log when they are run. From the session log display, you can take further actions, such as displaying more information on the messages in a log.

Reading The Session Queue

The information in the session queue includes the following:

- *Creator*: The user who generated the session.
- *Date and Time*: The date and time when a session was generated (entered in the session queue).
- *Lock date*: If a session is locked, this column shows the date upon which the system will release the session. A locked session cannot be started.
- *Authorizations user*: The user under whose authorizations the transactions in a session are to be run. When the session is run, it can execute only those transactions and functions for which this user has authorizations.
- *Status*: The status of a session (new, processed, contains errors, as shown in the tabs in the overview).
- *Tran.* and *Screen*: The number of transactions and screens, respectively in a session.

You can display statistics on the transactions in any session by marking the session and using the *Statistics* function.

Session Sorting and Status

Sessions in the session queue are sorted by date and time of generation and are grouped in different lists according to their status.

Selecting Sessions and Reading the Session Display

Processed	Session processed successfully
Incorrect	Session processed but contains transaction with errors
New	Session was recorded but has not processed yet
Being recorded	Session is being generated
Being processed	Session is being processed
Batch	Session is scheduled for batch processing

Possible statuses are as follows:

- new and not yet processed
The *Tran.* and *Screen* fields in the display show how many transactions and screens, respectively, the session contains.
- held in the session queue because of errors in transactions (Errors in sessions)
Transactions that contained errors are aborted; all correct transactions are processed. The *Tran.* and *Screen* fields in the session display show how many incorrect transactions were found in the session.
You can restart a session and correct the erroneous transactions with one of the interactive execution modes offered by the batch input system. For more information, please see [Correcting a Session \[Page 196\]](#).
- processed
For further information on a session that has been successfully run, you can display the log generated by the session. All completed sessions generate a log. You cannot run a completed session a second time.
Only sessions that were generated with the KEEP option are held in the queue after processing. Other sessions are deleted after they are successfully completed.
- in generation
You will usually see this status only if you happen to display the queue while a session is being generated (entered into the session queue).
You can also encounter this status if a system failure has interrupted the generation of a session. If you suspect that a session has been interrupted, please see [Releasing and Restarting an Interrupted Session \[Page 202\]](#) for more information.
- in process
You will usually see this status only if you happen to display the queue while a session is being run.

Selecting Sessions and Reading the Session Display

You can also encounter this status if a system failure has interrupted the execution of a session. If you suspect that a session has been interrupted, please see [Releasing and Restarting an Interrupted Session \[Page 202\]](#) for more information.

Starting Sessions Explicitly: Run Modes

Starting Sessions Explicitly: Run Modes

Use

Running a batch input session executes the transactions in the session and enters data into an R/3 System.

Usually, the system will run batch input sessions automatically. However, you can also start batch input sessions by hand. You may wish to do this for these and other reasons:

- To correct transactions that had errors
- To check that the transactions in a session have been generated correctly by running the first several transactions
- To start a session on special request (the session would not be started automatically or must be started right away).

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

To start a session, mark the session and choose *Process* from the tool bar. You can then choose how to run a session and with what logging and display options.

You can start any session in *new* status that is not locked. With *Process/foreground* or *Display errors only* mode, you can also re-start transactions that have the status *Incorrect*. Sessions with the status *Processed* cannot be run again.

Run Modes

There are three ways to run a session:

Process on screen	Display all dialog steps
Display errors only	Display dialog steps only for error or termination messages
Process in the background	No display of dialog steps, background processing

- **Process/foreground:** You can interactively correct transactions that contained errors and step through transactions that have not yet been executed.
- **Display errors only:** This mode is like *Process/foreground* except that transactions that have not yet been run and which do not contain errors are run non-interactively.

Starting Sessions Explicitly: Run Modes

If an error occurs, processing stops and the screen upon which the error occurred is displayed.

- **Background:** Use this mode to schedule a session for immediate processing in the background processing facility.

You receive control of your terminal again as soon as the session has been passed to the background processing system.

Note that your session is automatically released for processing in the background processing system. You need not explicitly release the session for processing.

A completed session is handled in one of three ways:

- The system deletes a session from the queue when it has been successfully completed. You can check on the outcome of the session by displaying the session log file.
- If the KEEP option was set when the session was generated, then the system leaves a session in the queue, even if it was run successfully. The status is changed to *Processed*.

You cannot run a processed session a second time. You must manually delete it when you no longer need it.

- If a transaction in the session contained an error, the incorrect transaction is aborted. All other transactions are completed. The session remains in the queue and is held in the *Errors* section of the list. You can correct the session in one of the interactive modes and run it to completion.

A transaction contains an error if it issues a message of type E (error) or type A (abnormal termination). Other messages are ignored and do not affect the execution of a session..

A session also is held in the queue in the *Errors* list if the session was ended with the /bend OK code. Please see [Correcting a Session \[Page 196\]](#).

For more information on correcting sessions with the display-all or error-display mode, please see [Correcting a Session \[Page 196\]](#).

Analyzing Sessions

Analyzing Sessions

Use

Use the analysis functions to check on the actions performed by a batch input session. The analysis functions allow you to display the transactions, screens, and data in a session.

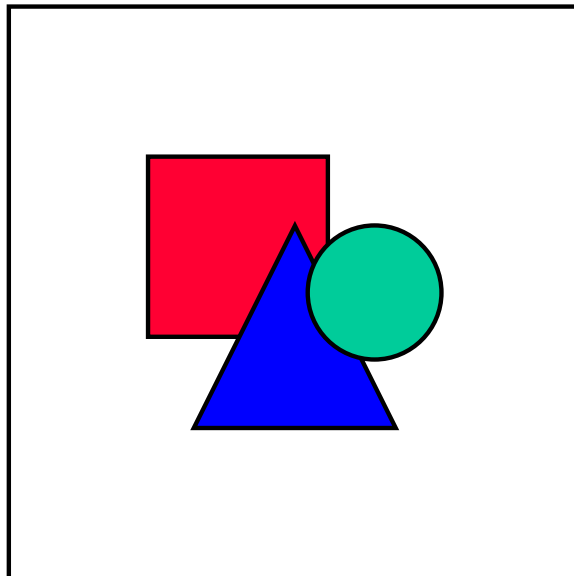
The analysis functions do not test a transaction or determine whether it will run correctly.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

You can use the analysis functions before or after a session has been run.



Analysis of sessions that have been run is available only for sessions that contained an error or which were generated with the KEEP option. The KEEP option prevents the system from deleting a session that has been successfully run.

You can analyze a session by marking the session in the session queue and selecting *Goto* → *Analyze session* or by selecting *Analysis* from the session log screen.

Statistics

For a quick summary of the size and contents of a session, you can use the statistics function: *Goto* → *Session statistics*. The advantage over the analysis function is that the summary is displayed more quickly.

Analysis Blocks

Reading very long sessions from the database can cause time-outs to occur. For this reason, the system reads sessions for analysis in blocks.

Blocks do not restrict your ability to display a long session. You can switch forward and backward from block to block with function keys.

A block always contains only complete transactions. A block boundary cannot fall in the middle of a transaction.

Session Summary

The *Transactions* analysis tab lists the transactions and screens recorded in the session.

Screens are identified by the name of the program and the number of the screen. Statuses are identical to those of the session queue, except that transactions that were deleted with the /bdel OK code have the status *Deleted*.

Session Data

You can display the data entered on each screen of a session by switching to the *Dynpros* tab.

Here, you can have the data presented to you in two formats:

- The first data display presents data in list format by field name. Choose this display by choosing *Options* from the tool bar and marking the *Field list* check box.
- Alternatively, you can switch to the screen called by the session with the session data shown on it. Double-click on the screen in the *Dynpro* list. The system simulates the screen. No data is processed.

You can switch back and forth from one form of display to the other.

Correcting a Session

Correcting a Session

Use

This section explains how to correct errors in faulty transactions in a session.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

When a session is run in background-processing mode, the system marks transactions that contain errors as incorrect. All other transactions in the session are completed. The session itself is kept in the session queue in the *Errors* section of the list.

A transaction contains an error if it generates an error message of type E (error) or type A (abnormal termination). Messages of other types are ignored and do not affect the execution of a session.

You can correct and re-execute the incorrect transactions in an "Errors" session. There are two ways to do so:

1. Re-run the session in display-all or error-display mode. These modes offer the following advantages:
 - The system skips transactions that were successfully completed. Only incorrect transactions or transactions that have not been executed are run.
 - You can change inputs and add missing screens interactively as incorrect transactions run. The system logs all such changes.

The following topic provides more information on using display-all mode or error-display mode to correct a transaction.
2. As an alternative, you can analyze the session to determine what the error was. With the analysis displays, you can check the screen that contained the error and the values that were entered in it. You can then correct the program that generated the session and regenerate the session.

You must ensure that the new session does not include transactions that were successfully completed. Otherwise, the updates made by the transactions will be performed again.

Using Display-All Mode

When you use display-all mode to re-start a session that contains an incorrect transaction, the system skips to the first screen of the incorrect transaction. Completed transactions are not re-executed.

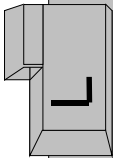
Options When You Run a Session Interactively

Change or enter data on screens

Branch to other screens or functions in the same transaction and then return to the next screen in the sequence defined in the session

Terminate a transaction

Delete a transaction



Process the screens in a session one after another with the data defined in the session shown on the screens (Enter key)

To correct a transaction, you can:

- use the **ENTER** key to step through the screens recorded in the session
As in normal operation, pressing the **ENTER** key starts a dialog step. As long as you simply press **ENTER**, the system processes the screens and data recorded in the session.
- change or add values in any screen
When you press **ENTER**, the system processes the data as altered by you. All of your changes are logged; you can review them at a later date if required.
- branch off from the transaction flow recorded in the session
You can use any function that you wish within the transaction that is currently running. You can, for example, branch off to enter data on a screen that was omitted from the session.
To resume executing the session, return to the screen that was expected by the session.

Correcting a Session

It holds the data for the screen recorded in the session until you have returned to that screen. It then resumes normal execution of the session.

Interrupting a Session

You can interrupt the interactive execution of a session by entering the **/bend** OK code on any screen. **/bend** terminates the transaction currently being executed in the session and marks the transaction with the status *Incorrect*. The session is kept in the queue and is displayed in the *Errors* section of the list. Changes made by the interrupted transaction are rolled back as long as the transaction uses only the R/3 update facility. Direct database changes made by the transaction are not rolled back.

You can use **/bend** when testing sessions. For example, you may wish to run the first transactions of a large session in display-all mode to make sure that the session has been generated correctly.

If the transactions are correct, you can then terminate the run with **/bend** and then submit the session for background execution. The transactions that have already been run will not be run again.

Restarting a Transaction

You can restart processing of a transaction by entering the **/bbeg** OK code on any screen. **/bbeg** terminates the transaction that is currently being processed and then restarts the transaction fresh, as it is recorded in the batch input session. Any changes made by the transaction are rolled back, as long as they were made only by way of the R/3 update facility. Changes made directly to the database are not rolled back.

Deleting a Transaction

You can delete a transaction from a session during interactive execution by entering the **/bdel** OK code. The transaction is removed from the session. The deleted transaction cannot be run even if you restart the session, and you cannot take back the deletion.

The transaction is, however, available for analysis if the session was generated with the **KEEP** option set. The transaction is marked with the status *Deleted* in the analysis display. The deletion also is recorded in the log generated by the session.

Correcting a Session

Function	OK Code
Terminate current batch input and mark as incorrect	/n
Delete current batch input from session	/bdel
Restart a transaction	/bbeg
Terminate batch input processing and mark session as incorrect	/bend
Change display mode to process the on screen instead of displaying only	/bda
Change display mode to display only instead of processing the sessions on	/bde

Result

Correcting and re-running faulty transactions in a batch input session lets you finish the processing of a session and complete the data transfer.

Deleting Sessions

Deleting Sessions

Use

Normally, batch input sessions are deleted from the session list automatically when they are completed. However, sessions remain in the list if any of the following is true:

- They were generated with the KEEP option.
- They contain errors or were aborted with the /bend OK code. (Such a session is deleted when you finish processing it, unless it was generated with the KEEP option.)

When you no longer need sessions that have been held in the list, you can explicitly delete them.

Do not delete any sessions that contain unprocessed transactions. You must first correct and process such transactions or enter the data in the transactions into the R/3 System in some other way.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system.

Procedure

To delete a session from the session queue, mark the sessions to be deleted and choose *Delete* from the tool bar. The system asks you to confirm the deletion. It also asks you if the session log should be deleted together with the session.

Locking and Unlocking Sessions

Use

You can lock a session to prevent the system from running it before the date that you specify in the lock.

For example, a session locked until the eleventh of November can be started only after that date.

You may wish to lock a session for such reasons as

- holding a session until a specified date
- preventing a session that contains errors from being re-started
- preventing a session that requires a special resource, such as printer forms or a specific tape, from being started unintentionally.

You can unlock a session by changing the lock date to the present date or by entering a space as the lock date.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

You can lock sessions in the session queue by marking the sessions and choosing *Lock* from the tool bar.

Releasing and Restarting an Interrupted Session

Releasing and Restarting an Interrupted Session

Use

If a system problem occurs while a session is being generated or is being run, the session is terminated abnormally. Typical causes of an abnormal termination might include shutdown of the R/3 System while a session is running or termination of a SAPGUI presentation process while a session is being run interactively.

In the session queue, a session that was terminated while it was being generated remains in the *Being generated* tab in the overview. A session that was terminated while it was running remains in the *Active* tab.

To prevent such sessions for remaining in these statuses indefinitely, you must intervene. You must release and either restart or delete and recreate these sessions.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

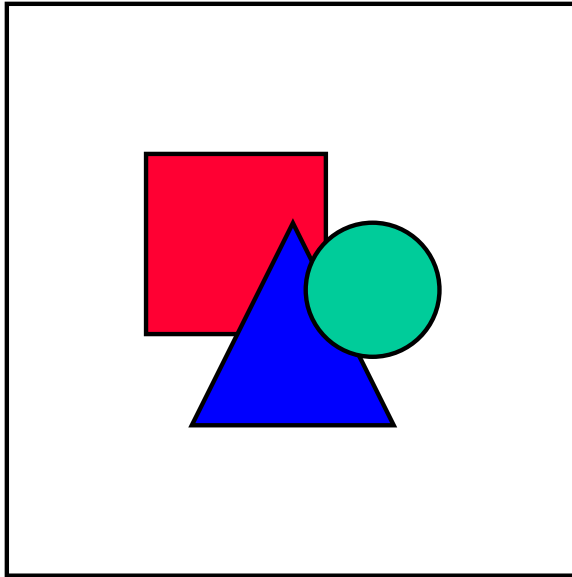
You can recognize that a session has terminated abnormally because the status of the session does not change. If, for example, you find a session that was run the previous night still in the *Active* list, then it is likely that the session was interrupted.

If a session was to run in the background, you can check the status of the session's job in the background processing system. If the session was aborted, then the background job log contains the reason for the abnormal termination.

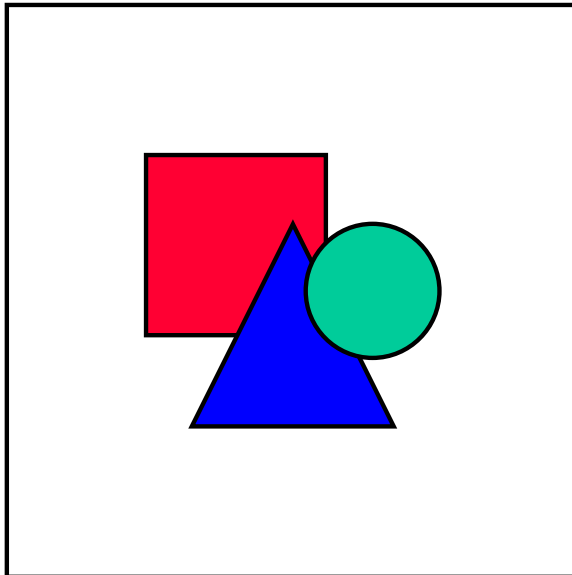
Before you can restart (or start) a session that terminated abnormally, you must release it. Releasing the aborted session sets its status to *Being processed*. Reset the status by marking the session or sessions and selecting *Release* in the toolbar.

You can safely restart a session that was interrupted while it was being run. When you restart the session, all transactions that were successfully completed before the problem occurred will be skipped. The transaction that was executing when the problem occurred will be re-executed.

Releasing and Restarting an Interrupted Session



Release an active session only if you are sure that it has been interrupted and is no longer running.. Sessions also have the status *Active* while they are running.



Restarting after abnormal termination during generation: You can start a session that was interrupted while it was being generated. The session is guaranteed to contain only complete transactions and to be runnable.

However, there is no guarantee that the session contains all of the transactions that were to be included in it. You may therefore wish to delete and regenerate such a session.

Displaying Queue Management Information

Displaying Queue Management Information

Use

Batch input sessions use the generic queue management system of R/3. If required, (for example on request by SAP Service), you can display a dump of a queue that is being used for batch input. The dump includes queue management information as well as the management information and data in the session.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions*. Alternate: Enter transaction SM35. Both paths take you to the session overview of the batch input system. transaction SM35.

Procedure

To display queue information, mark a session and select *Goto* → *Queue*.

Not all of the queue management fields displayed are used by the batch input system; the queue management functions are used by other queues in the system as well. Some of the most important fields that are used by the batch input system are as follows:

- *Client*: The client in which a session is to run.
- *Groupid*: The session name.
- *QID*: The internal ID of a session. The ID is used for queue management with *System* → *Services* → *Queue*.
- *QSTATE*: A character indicating the status of the batch input session.
- *QERASE*: If marked, indicates that the KEEP option is set. If KEEP is set, the system does not delete the session after it is successfully run.
- *USERID*: User to be used for authorization testing if a session is submitted for background execution.

Displaying Session Logs

Use

The batch input system keeps a detailed log of each session that is processed. The log contains not only progress messages from the batch input system itself, but also error messages from the transactions that are processed.

To analyze an error in a session, you should start by checking the session log for relevant messages.

A session log is kept only if the session was generated with the KEEP option or if the session is aborted or contains an error.

Prerequisites

Start the batch input management tool: Select *System* → *Service* → *Batch input* → *Sessions or Logs*. Alternate: Enter transaction SM35P. You can display logs from the standard session overview, or from a special separate transaction for logs. The log functionality is in each case the same.

Procedure

To display a log, mark a session and choose *Log* from the tool bar.

A session log contains any error messages issued by transactions in the session. It also includes batch input error messages reporting any problems in running transactions, together with the transaction and screen at which the problem occurred. Finally, a log contains a set of summary statistics.

Reorganizing the Batch Input Session Log File

Reorganizing the Batch Input Session Log File

Use

You should periodically use the ABAP/4 program RSBDCREO to reorganize the batch input log file. You can run RSBDCREO in the background or interactively.

Running the program reduces the size of the log file, BI<R/3 System name><Instance name>, in the shared R/3 directory "global" in the host system to the minimum possible size. Deleting a session marks a session log for deletion, but the storage held for the log is returned to the host only by a reorganization.

The program will delete a log only if the session to which the log belongs has been deleted.

Procedure

Schedule RSBDCREO as a job in the background processing system. You will find information on the job specifications for RSBDCREO in the background processing documentation in the CCMS Guide.

Maintaining Tables

Tables are a central component of the R/3 System. They can be distinguished according to the type of information they contain and can both manage data and carry out control functions.

This section describes table attributes and explains how tables are maintained.

[Overview of Tables \[Page 208\]](#)

[Client-Specificity: Maintaining Tables \[Page 210\]](#)

[Maintaining and Displaying Table Contents \[Page 211\]](#)

Overview of Tables

There are four main types of table:

- **Tables containing system control data**

These tables fulfill technical control functions within the R/3 System. They are maintained either by SAP or by users with special authorization.

- **Tables containing basic commercial data**

These tables contain data such as postal codes, country keys, and wage types and are filled with default values by SAP. You will need to check these tables and maintain them where necessary. For example, different clients may require different control logic.

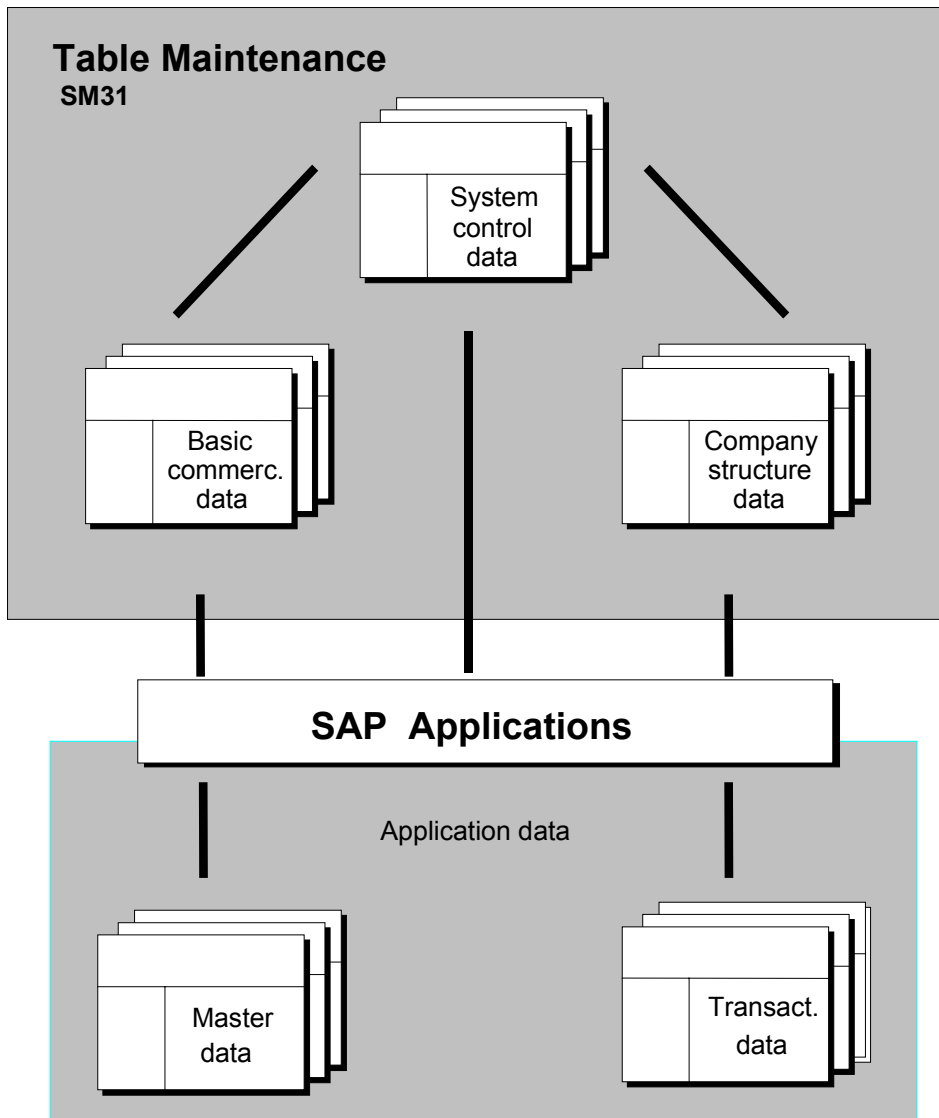
- **Tables containing data on the company structure**

These tables contain data such as company codes, plants, and storage locations and customer-specific control data such as printer information and authorizations. These tables are partially filled with sample values by SAP. You must fully maintain these tables.

- **Tables for application data**

These tables contain transaction and master data and are processed using R/3 applications.

This diagram shows the interaction of tables in the R/3 System.



Client-Specificity: Maintaining Tables

Client-Specificity: Maintaining Tables

A table can be either client-specific (applying only to a single client) or client-independent (applying to all clients in the R/3 System).

- **Client-independent tables**

Client-independent tables contain data of general relevance, for example, R/3 system control data, language indicators, and transaction codes.

- **Client-specific tables**

Data which only applies to one client is stored in client-specific tables. Examples of client-specific tables are tables containing application data, tables containing information about the structure of a company, and, with a limited number of exceptions (for example, language indicators), tables containing basic commercial data.

A table attribute in the Data Dictionary indicates whether a table is client-specific or client-independent (see also "Creating a Table Structure" in the *ABAP/4 Dictionary* guide). In client-specific tables, the client is always entered in the first key position.

Maintaining and Displaying Table Contents

Tables containing master data and transaction data are always processed by R/3 applications. All other tables can be maintained directly with the standard or extended table maintenance functions. Extended table maintenance is described in the next section. If you have installed the ABAP/4 Development Workbench, you can also display and edit table contents with the Data Browser, found in the *Test* menu.

The extended table maintenance function offers extra features, such as marking of blocks of entries, selection by field contents, and a "waste paper basket" buffer for deleted entries. The standard table maintenance function makes changes directly in the database, without buffering them. The standard table maintenance function also accepts only table names that are five characters long or less. Because of the extra comfort of the function, SAP recommends that you use it rather than the standard table maintenance.

Selecting the Maintenance Interface

The standard table maintenance function will be replaced by the extended table maintenance function in a future release. You can already use the interface of the extended table maintenance function in the standard function.

If you create a new table, you can decide which of the two interfaces you wish to use and then create the appropriate interface.

For existing tables, only the standard interface is available in the standard maintenance function.

Creating a Standard Maintenance Interface

To create interface screens for a new table using the interface of the standard table maintenance function, do the following:

1. Select *Tools* → *Case* and then *Development* → *Screen painter* → *Utilities* → *Create table screen*.
2. Enter the name of the new table and press *Continue*.

Starting the Standard Maintenance Function

You can maintain or display tables with the menu options

System → *Services* → *Table Maintenance*. You can display or maintain table contents. If the table you select already has the extended interface, the System automatically starts the extended maintenance table function for you.

Authorizations

Access to tables is controlled by means of authorizations. The authorizations check type of access (display and maintenance) and type of table (client-specific or client-independent).

Searching for Table Entries

Starting from the left, enter as much information as you can into the search fields under the table display. When you press ENTER, the table is displayed from the first line matching your search data.

Maintaining and Displaying Table Contents**Adding and Changing Table Entries**

You can add or change table entries in the following ways:

- add or change an entry by overwriting at least the key field of an existing entry.
- type a new entry into an empty line at the end of the table.

When you press ENTER, the system checks the key data you have entered against the existing key data. If the R/3 System does not find another key that matches the one that you have entered, then the new entry is added. The entry which has been overwritten is not affected by this.

If the key data is identical to that of another entry, the existing entry is replaced by the new one.

Deleting Table Entries

Position the cursor on the entry you wish to delete and select the delete function.

Extended Table Maintenance

[Overview \[Page 214\]](#)

[Concept \[Page 215\]](#)

[Call And Operation \[Page 220\]](#)

[Generate Extended Table Maintenance Dialog \[Page 218\]](#)

[Processing Mode \[Page 221\]](#)

[Authorizations \[Page 217\]](#)

[Create Sub-Work Areas of Records \[Page 222\]](#)

[Display Functions \[Page 226\]](#)

[Maintenance Functions \[Page 227\]](#)

[Transport \[Page 230\]](#)

Overview

Overview

The extended table maintenance offers a convenient standard maintenance dialog for processing table contents, which can be modified for use in applications.

The maintenance dialog can be used for table views as well as for pure tables (without view definitions in the Dictionary) of type "C" (maintenance) or "H" (help).

It will therefore gradually replace the existing table maintenance, which only remains valid for tables which were created before Release 3.0. Such tables often do not yet contain the maintenance modules which are required for the extended table maintenance. For such tables, the maintenance modules can be generated later, so that the extended table maintenance can also be used for them. Other tables have individual modifications in the maintenance logic which must first be integrated in the extended table maintenance logic. Such old tables are currently being converted by SAP.

The maintenance dialog has comprehensive user interface functionality, which guarantees that maintenance procedures are comprehensible and transparent.

The extended table maintenance dialog can be called via the menu or be integrated into applications.

Concept

You can conveniently maintain table contents in a standard maintenance dialog with the extended table maintenance. It is irrelevant whether you access the data via a table or a view. You can work with the same interface and functionality in both cases.

The generation of maintenance modules for the table or view in question is a prerequisite. The generation can be called for the current table or view from within the Workbench or directly in the Dictionary via a function.

To call the maintenance dialog for a particular table or view, either a specific transaction must have been defined, with which you then go directly to the maintenance, or you call the general maintenance transaction and specify the table or view name. If modifications were made in the maintenance dialog for a table or view, they apply however you entered the maintenance.

Internal Processing

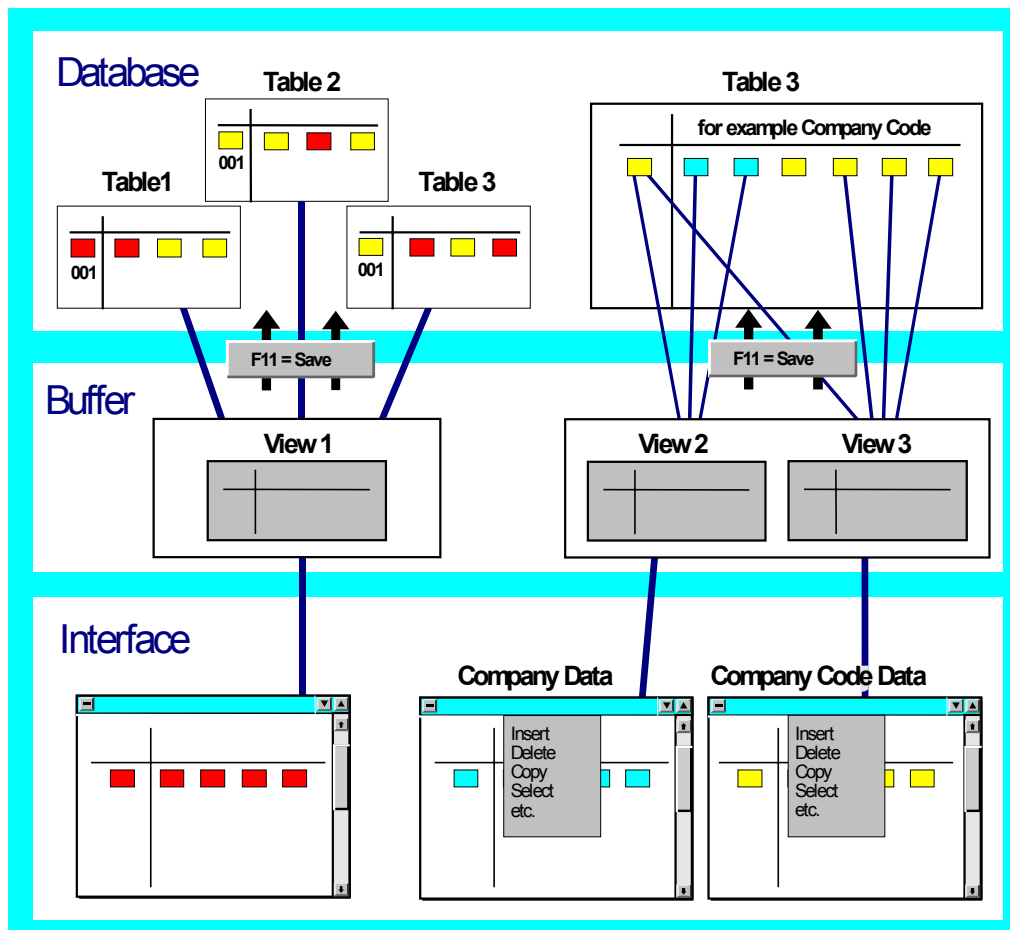
When the extended table maintenance is called, a work area is fetched from the database into an internal buffer for processing on the screen. Either all entries in the database are fetched into the work area, or the selection can be restricted in a view definition or in the calling program.

Field contents are maintained in an internal buffer, so the database is not automatically accessed after each individual record has been maintained. Changes made are only copied from the internal buffer to the database when save is chosen. This buffering gives the maintenance procedure a transaction orientation. This gives you the possibility of discarding changes before the database access (user-controlled "rollback") and allows consistency checks for all changed data, before it is written to the database.

The following diagram illustrates the extended table maintenance using two views as examples:

Extended table maintenance

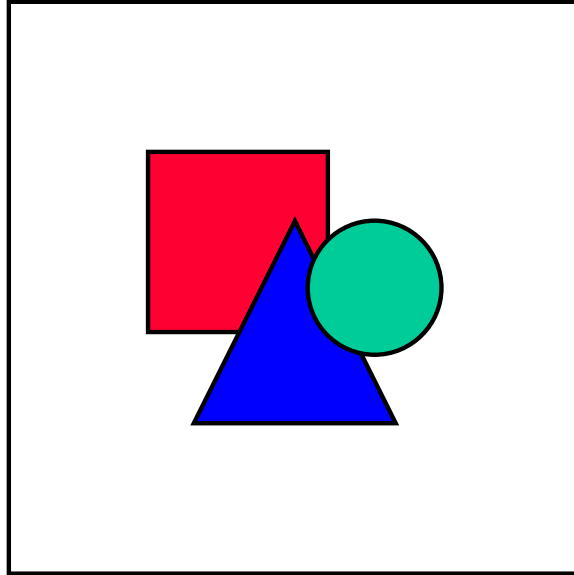
Concept



Authorizations

When client-independent tables are maintained, the global authorization object “Maintenance of client-independent tables” is checked to see whether authorization for maintaining client-independent tables is granted.

The authorization object “Table maintenance” controls whether a user may maintain or display the data in a particular table or view. This authorization is controlled via the authorization group which is specified when the table or view maintenance modules are generated.



You can find further information on this topic in the system administration documentation, section “Authorizations”, “Table and view maintenance”.

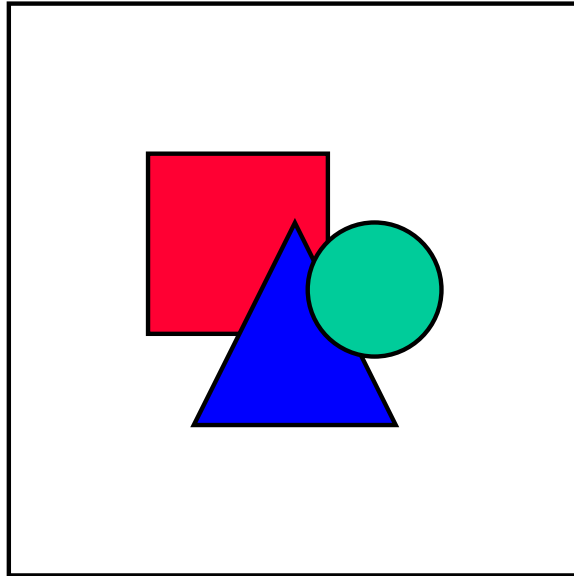
Even if authorization exists, a field can only be maintained if the field is also defined as maintainable in the Dictionary.

Generate Extended Table Maintenance Dialog

Generate Extended Table Maintenance Dialog

To generate the maintenance modules for a table or view, proceed as follows:

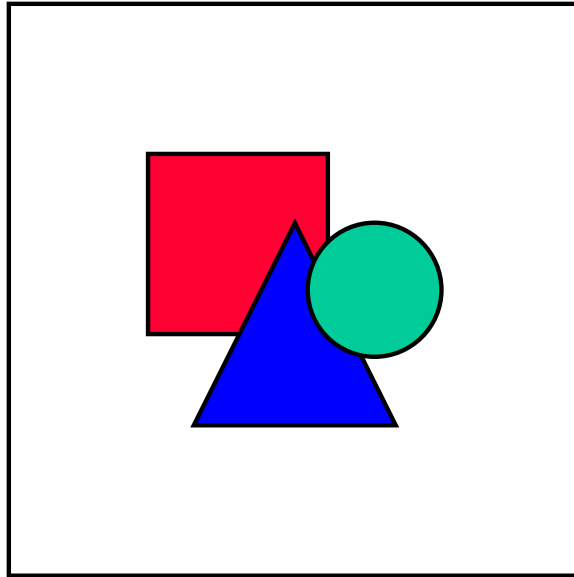
1. Choose *Tools* → *ABAP/4 Workbench* → *Development* → *Other tools* → *Gen.tab.maint.dialog*. You arrive in the maintenance transaction initial screen.
2. Enter the name of the table or view.



Alternatively to steps 1 and 2, you can call the function *Utilities* → *Gener.maint.dialog* for the table or view in question in the Dictionary (*Tools* → *ABAP/4 Workbench* → *Development* → *Dictionary*). You arrive directly in the maintenance screen for the current table's generated objects.

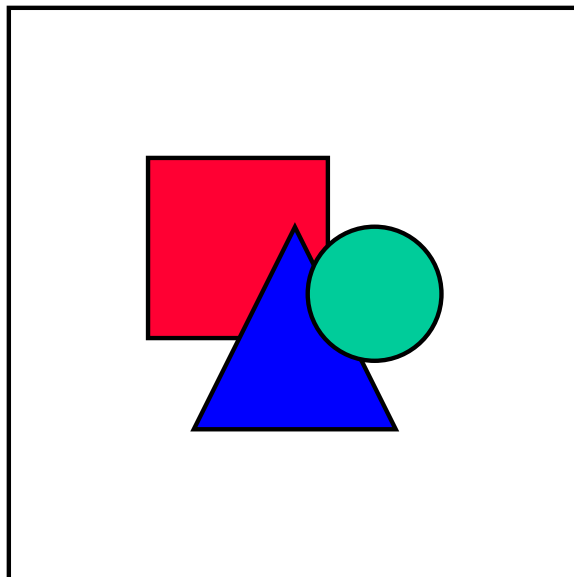
3. Choose the option "Generated objects".
4. Choose *Create/Change*.
5. Confirm that the maintenance modules are to be created in the next dialog window.
6. Enter the data required for the generation:
 - Function group to which the maintenance modules are to belong

Generate Extended Table Maintenance Dialog



One function group can contain maintenance modules for several tables or views.

- Authorization group
 - Maintenance type (one or two-step)
 - Maintenance screen(s) (one or two-step maintenance type, resp.) numbers
 - Recording routine (Standard/individual or none)
7. Choose *Create*. All required maintenance modules are now generated.



If you want to make changes later, you must call the function *Change*, to regenerate the maintenance modules in question.

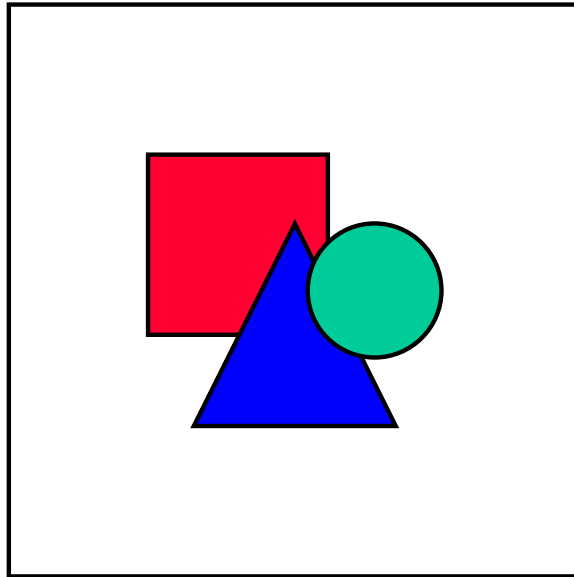
Then you can process the table with the extended table maintenance.

Call And Operation

The maintenance dialog can be called in the following ways:

- Call the complete **maintenance transaction** (from the application or via the system menu)

When calling the maintenance transaction, you can restrict the records which will be read from the database for display and processing. The menu call selection path is *System* → *Services* → *Ext. table maint.*



If the records read from the database were restricted, you can also only maintain, including create, records which satisfy the restrictions.

This restriction is not the same as selecting among records for particular processing (e.g. delete or copy) during processing.

- Call the **maintenance function modules** (only possible from an application)

The application can control maintenance processing. This control could be e.g. a restriction of the work area which is read in from the database, or a data consistency check before writing to the database.

An overview (list) screen for the table or view which is to be processed is displayed first. For two-step procedures, or tables or views whose records contain more fields than can be displayed in one screen line, a detail screen can be called for the individual overview screen records.

Processing Mode

In the maintenance transaction initial screen you can choose between three processing types. At the same time, you can specify whether you want to work with all the database records, or whether you want to make a selection. To restrict the selection, select the option "restricted data area" before you choose a processing function. After you have fetched a complete or restricted work area into the buffer or onto the screen for processing, you can either process individual records or create a sub-work area.

The processing modes in the initial screen are:

- **Maintain** (Change, Insert, Delete entries)

Maintenance processing can also be restricted by field, i.e. you may have authorization, but if you try to maintain a field which is flagged as "Read only" in the Dictionary, it will not be ready for input.

- **Display**

In this processing mode you can only display. It is possible to select by field contents to create sub-work areas. Maintenance functions can not be used.

- **Transport**

Changed records are usually copied into a change request when they are saved.

"Usually" means (see also [Transport \[Page 230\]](#)):

- for client-independent tables or views
- for client-dependent tables or views, if the client is set.

If this is not the case, or if you want to transport unchanged records for other reasons, you can manage records in change request tasks in this mode.

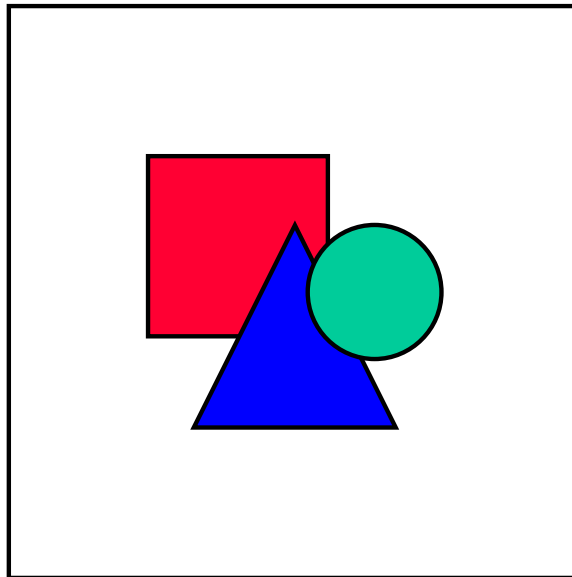
Create Sub-Work Areas of Records

Create Sub-Work Areas of Records

As well as processing individual records, you can also group a selection of records for processing. All records which satisfy the specified selection conditions are displayed and can be processed. This is useful e.g. when copying all processed records into a transport task.

Selection by Field Contents

1. Choose *Selection* → *By contents* in the maintenance screen. You get a list of all fields used.
2. Select the desired field or fields.
3. Choose *Continue*. You get another screen, in which you can enter the comparison value for the field or fields.
4. Enter the comparison value.



The compare operator has the default value "=" (equality). You can display the possible compare operators for values or character strings with F4.

5. If you want to extend the selection, display the field list again with the function *Append*, select the desired field or the fields and choose *Continue*.

If you want to add a selection condition, position the cursor on the line **before** the one where you want to make a selection. With the function *Insert* you get the field list, from which you can select the desired field.

You can link the selections with a logical "AND" or a logical "OR". Each OR operator forms a logical search block, which can consist of several AND statements. These search blocks are visually separated when you select ENTER.



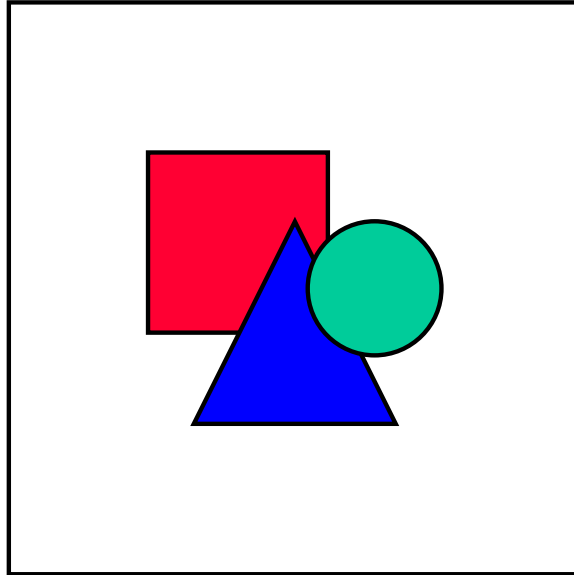
Field1 = 001 AND (color1)

Create Sub-Work Areas of Records

Field2 = xyz OR (color1)

Field1 = 002 AND (color2)

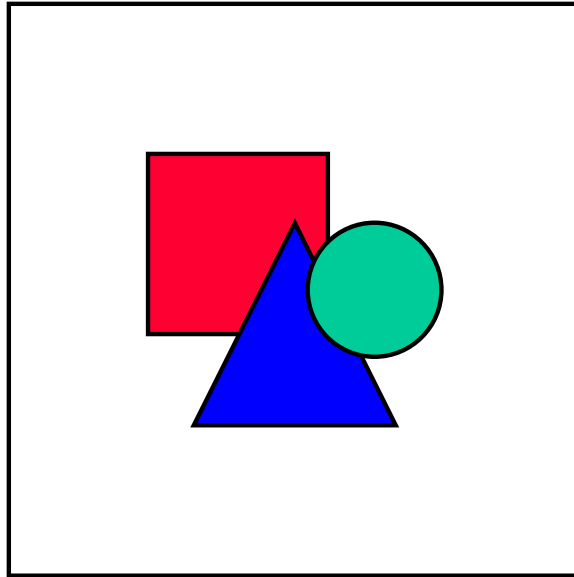
Field2 = abc (color2)



You can process, e.g. move or even delete the selection conditions by selecting the line in question and calling the appropriate function. You can see the available processing functions by pressing the right-hand mouse key in the dialog box. You delete a single condition e.g., by selecting it and then choosing *Delete*. You can delete all conditions via the function *New selection*.

6. The function *Choose* displays all records which satisfy the specified selection conditions, for maintenance.

Create Sub-Work Areas of Records



The escape symbol "#" can be used for case-sensitive searches

Select Functions

You can select single records, all records or a block of records. You can display all selected records together (*Choose ☐ → ☐ All selected*) and process them.

You can *select* any number of entries on the overview screen. If you are in a detail screen, this record is handled like a *selected* record.

You can *select* several records in the following ways:

- **Select single records**

Select the desired records by mouse click on the select box or F9.

You can get an overview of all selected records with the menu function *Choose ☐ → ☐ All selected*.

- **Select all records** of the entire work area or sub-work area (e.g. all changed records).

Choose *Process ☐ → ☐ Selecting ☐ → ☐ Select all*.

- **Selecting a block** of records

Position the cursor initially on the first record in the block and choose the menu option *Edit ☐ → ☐ Selections ☐ → ☐ Select block*, then position the cursor on the last record in the blocks and repeat the *Select block* function.

To delete a selection, use one of the two following possibilities:

- **Delete single selection**

Position the cursor on the selected record and repeat the select function (see above).

- **Delete all selections**

Use the menu option *Edit ☐ → ☐ Selections ☐ → ☐ Delete all selections*.

Create Sub-Work Areas of Records

You can perform all maintenance or display functions with the selected records (see the relevant topics below).

Select Processed Records

All processed records are stored in an internal buffer until they are saved, and they can be changed and retrieved at will. This functionality includes all inserted and deleted records. You can display processed records in this way, to check, and reverse if necessary, their changes.

You can choose the following record selections in the *Selection* menu:

- **all changed**
- **all inserted**
- **all deleted**
- **all in a task** (only in transport mode)
This selection is not affected by saving.
- **all not in a task** (only in transport mode)

This selection is not affected by saving.

Your entries and changes are only written to the database when you save. They are then, with the exception of the last two selections (transport mode) no longer available to the selection functions.

Display Functions

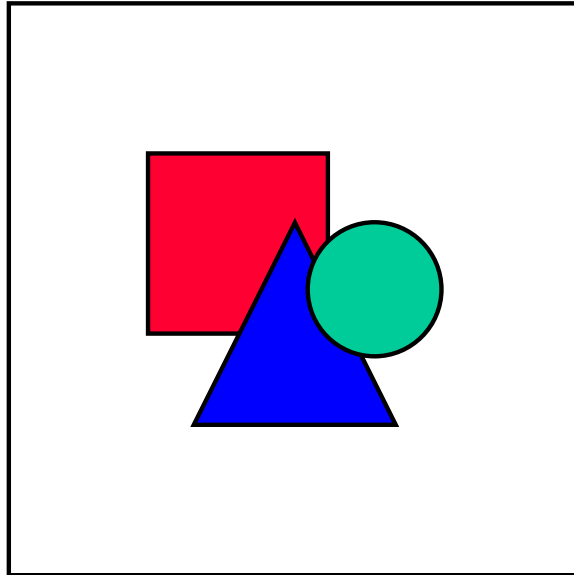
Display Functions

Apart from the display possibilities in the selection functions, described above, the following display functions can be found under the menu point *Goto*:

- *Next entry*
In the overview screen, the records displayed are shifted up one line, in the detail screen, the next entry from the work area (overview screen) is displayed.
- *Previous entry*
In the overview screen, the records displayed are shifted down one line, in the detail screen, the previous entry from the work area (overview screen) is displayed.
- *Other entry*
This function is only active in the overview screen. You can enter the key of a record. The overview screen is shifted until the entry with the specified key appears in the first line.

Maintenance Functions

You can only use the maintenance functions in the processing mode **Maintain**. The maintained records are at first stored in a buffer. The records are only written to the database when the function **Save** is called. This makes maintenance functions which recover previous status possible:



The recovery functions can only be used until the function **Save** is called.

- **Insert record (insert without reference)**

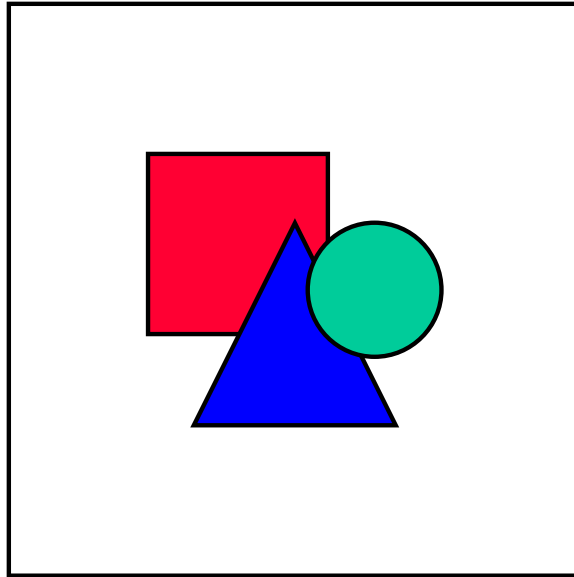
The function **Edit** → **New entry** enables you to maintain a new entry. In one-step procedures (only overview screen) you get an overview screen with empty lines for your input, in two-step procedures (overview screen plus detail screen), an empty detail screen.

- **Copy records (insert with reference)**

If you want to create new records whose field contents are to a great extent the same as already existing records, you can use the copy function. You can copy one or several records for processing, and change the fields as you want to save them.

- a) Select the record or records which you want to copy, and choose the function **Edit** → **Copy as...** You get the selected records, with all fields displayed ready for input.
- b) Make the desired changes to the copies.
- c) Insert the changed records as new records in the local buffer with **ENTER**.

Maintenance Functions



These new records are only written from the internal buffer to the database when the function *Save* is called, as for all other changes, and they can be discarded until then.

- **Delete records**

You can delete records individually or in blocks from the overview screen or the detail screen. You can also select by field contents and mark the records in this selection list for deletion:

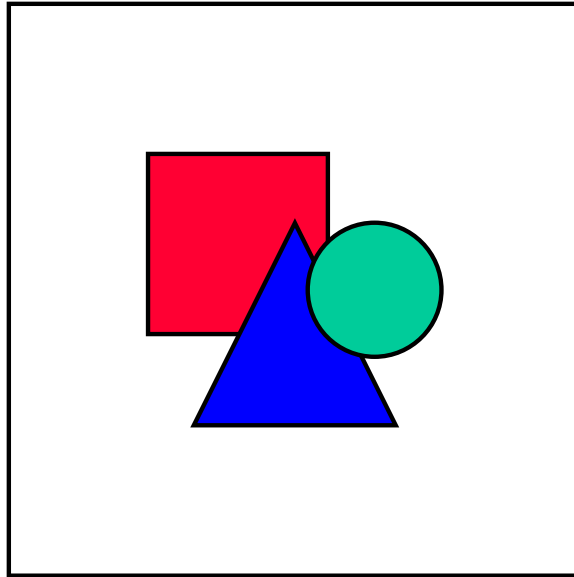
- a) Select the records to be deleted, individually or by blocks (possibly after a selection).
- b) Choose *Delete*.

- **Recover deleted records**

- a) Choose *Selection* → *Display deleted*, to display all deleted records on the screen.
- b) Select the record or records whose deletion you want to cancel, and confirm the function *Retrieve*.

- **Change/replace field contents**

With this function you can give a field the same contents in several records at the same time. This applies only to non-key fields.



If you want to change key fields, save the change as a copy, and then delete the source record.

- a) Select the records whose fields you want to change at the same time, and choose the function *Edit* → *Change field contents*. A dialog box with the field list appears.
- b) Select the field in which you want to make the changes and choose *Continue*.
- c) Enter the value which the field is to have in the previously selected records and choose *Replace*. The field in question takes the specified value in all selected records.

- **Recover original records**

You can recover the original status of a changed record.

- a) Select the record directly, or fetch a selection list of records with the function *Selection*, by marking one or several records.
- b) Choose the function *Edit* → *Recover original*.

Save

When you save, all maintenance functions (Change, Delete, Copy to correction, etc.) performed in the internal buffer since the maintenance was called, are written to the database.

You can thereafter no longer use the recovery functions or select by changed, deleted or inserted records.

It is, however, still possible to select records which are, or are **not**, contained in a change request task.

Transport

Transport

You must transport changed data to your productive system or productive client if you want to use them. When data changes are saved, you are usually automatically prompted by the system to enter a change request task number. This occurs in the following cases:

- For client independent tables or views, for which, according to the definition, the standard recording methods were not switched off and which do not belong to the delivery class "L" or "W".
- The client in question is set to "Record".

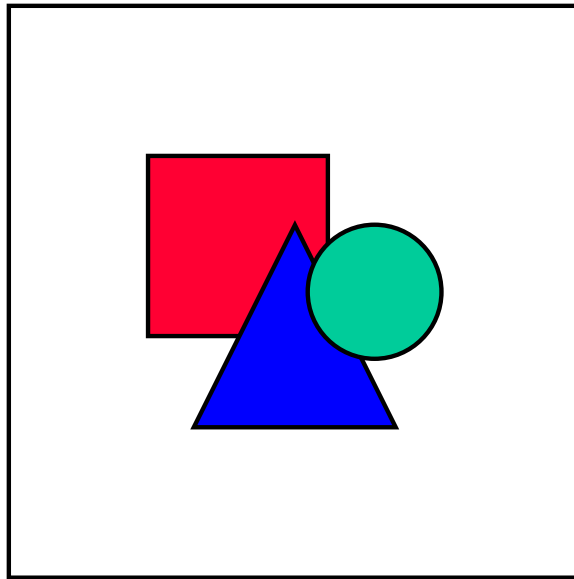
You can enter the number of an existing request in a dialog box or choose one from the F4 help list, or you can create a new request. The system then creates a task for the request. This task contains all records which are to be transported.

If you want to transport records independently of processing, you can put individual records or a selection of records in a task in the maintenance dialog. The following maintenance and display functions are available for this:

- **Put records in task**
- **Remove records from task**
- Display **all records in task**
- Display **all records not in task**

Put Records in Task

1. Choose either the pushbutton *Transport* in the initial screen, or the menu function *Table view* → *Transport* in the maintenance screen.
2. Enter a task number in the dialog box, or choose one from the F4 help list or the pushbutton *User requests*. Then choose *Continue*.
3. Select the records which are to be transported, and choose the function *Put in task*. The selected records are flagged for the task.



You can also first make a selection (e.g. by contents (menu *Select* → *by contents*) and then use the function *Select all*.

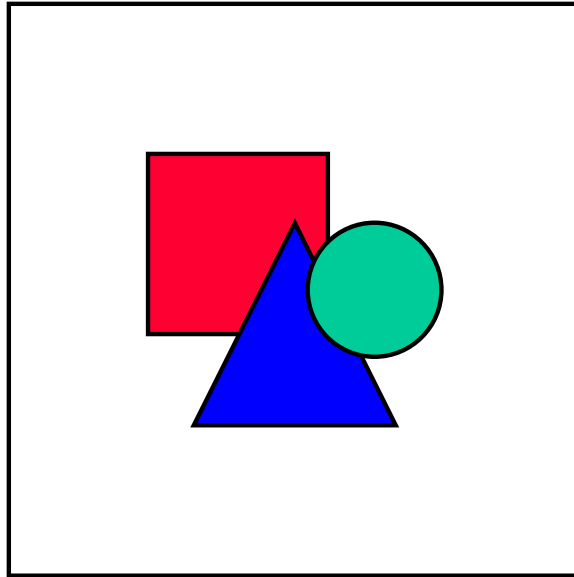
4. To actually put the selected records in the specified task, call the function *Save* (in the standard toolbar, with the F11 key, or *Table view* → *Save*).

Remove Records from Task

You can select records which you want to remove from the task again, just as you can select records for putting in a change task.

1. Choose either the pushbutton *Transport* in the initial screen, or the menu function *Table view* → *Transport* in the maintenance screen.
2. Specify a task number in the dialog box, or choose one from the F4 help list or the pushbutton *User requests*. Then choose *Continue*.
3. Select the records to be removed, and choose the function *Remove from task*. The records marked are flagged for deletion in the specified task.

Transport



It is useful here to use the function *Select* → *All in task*, to get a list on the screen of all records contained in the task. You can then check which records are contained in the task in question, and select those which you wish to remove.

Transport Records

With the above functions (*Put in task* and *Remove from task*) you put the records in tasks which are assigned to a change request.

After completing the changes, release the task to the associated request. Then release the request to transport into another system or another client. Perform these actions with the Workbench Organizer (In the R/3 menu: *Tools* → *ABAP/4 Workbench* → *Overview* → *Workbench Organizer*.)

Security Audit Log

Purpose

The Security Audit Log is a tool designed for auditors who need to take a detailed look at what occurs in the SAP System. By activating the audit log, you keep a record of those activities you consider relevant for auditing. You can then access this information for evaluation in the form of an audit analysis report.

The audit log's main objective is to record:

- Security-related changes to the SAP System environment (for example, changes to user master records)
- Information that provides a higher level of transparency (for example, successful and unsuccessful logon attempts)
- Information that enables the reconstruction of a series of events (for example, successful or unsuccessful transaction starts)

Specifically, you can record the following information in the Security Audit Log:

- Successful and unsuccessful dialog logon attempts
- Successful and unsuccessful RFC logon attempts
- RFC calls to function modules
- Successful and unsuccessful transaction starts
- Successful and unsuccessful report starts
- Changes to user master records
- Changes to the audit configuration

Implementation Considerations



The Security Audit Log contains personal information that may be protected by data protection regulations. Before using the Security Audit Log, make sure that you adhere to the data protection laws that apply to your area of application!

Integration

With the Security Audit Log, SAP Systems keep records of all activities corresponding to designated filters.

For a detailed description on the technical aspects of the audit log, see [The Design of the Security Audit Log \[Page 235\]](#).

The Security Audit Log complements the system log; however, the Security Audit Log has a slightly different purpose and a different audience (see [Comparing the Security Audit Log and the System Log \[Page 238\]](#)).

Security Audit Log**Activities**

For more information about the various activities that you need to perform when using the Security Audit Log, see:

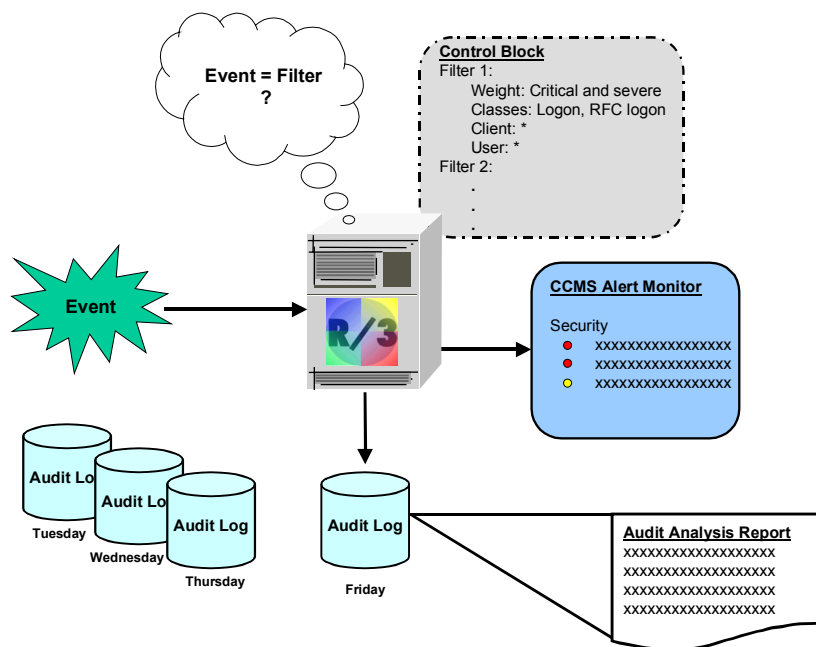
- [Defining Filters \[Page 244\]](#) to enable auditing and configure the information you wish to audit.
- [Displaying the Audit Analysis Report \[Page 246\]](#) for a detailed description on how to specify your audit analysis report. You can view the recorded information as desired. You can view everything that you have logged, or you can select a sub-group (for example, certain transactions or certain users).
- [Deleting Old Audit Files \[Page 250\]](#) for information on archiving and deleting your audit files.

The Design of the Security Audit Log

Overview

The Security Audit Log keeps a record of security-related activities in SAP Systems. This information is recorded daily in an audit file on each application server. To determine what information should be written to this file, the audit log uses filters, which are stored in memory in a control block. When an event occurs that matches an active filter (for example, a transaction start), the audit log generates a corresponding audit message and writes it to the audit file. A corresponding alert is also sent to the CCMS alert monitor. Details of the events are provided in the Security Audit Log's audit analysis report. See the graphic below:

Security Audit Log Architecture



SAP Systems maintain their audit logs on a daily basis. The system does not delete or overwrite audit files from previous days; it keeps them until you manually delete them. Due to the amount of information that may accumulate, you should archive these files on a regular basis and delete the originals from the application server (see [Deleting Old Audit Files \[Page 250\]](#)).

The Audit File / The Audit Record

The audit files are located on the individual application servers. You define the name and location of the files in the profile parameter `rsau/local/file`. When an event occurs that is to be audited, the system generates a corresponding audit record, also called an audit message, and writes it to the file. The audit record contains the following information (if known):

The Design of the Security Audit Log

- Event identifier (a 3-character code)
- SAP user ID and client
- Terminal name
- Transaction code
- Report name
- Time and date when the event occurred
- Process ID
- Session number
- Miscellaneous information

You define the maximum size of the audit file in the profile parameter `rsau/max_diskspace/local`. The default is 1000000 bytes (= 1 MB). If the maximum size is reached, then the auditing process stops.

Filters

You define the events you want to audit in filters. This information is stored in the control block, which is located in the application server's shared memory. The SAP System uses this information to determine which audit messages should be written to the audit file.

Filters consist of the following information:

- Client
- User
- Audit Class
 - Dialog logon
 - RFC/CPIC logon
 - RFC function call
 - Transaction start
 - Report start
 - User master change
 - Other
- Weight of Events to Audit
 - Only critical
 - Important and critical
 - All

For more details, see [Defining Filters \[Page 244\]](#).

The Audit Analysis Report

You can view the contents of the audit files in the audit analysis report. For more information, see [Displaying the Audit Analysis Report \[Page 246\]](#) and [Reading the Audit Analysis Report \[Page 248\]](#).

Alerts in the Computing Center Management System Alert Monitor

The Security Audit Log also generates security alerts for the events recorded in the Computing Center Management System (CCMS) alert monitor. For more information, see [Security Alerts in the CCMS Alert Monitor \[Page 251\]](#).

Comparing the Security Audit Log and the System Log

Comparing the Security Audit Log and the System Log

The Security Audit Log complements the System Log. Both are tools used to keep a record of activities performed in SAP Systems. However, they use slightly different approaches and have different objectives. We show how these two types of logging compare in the table below.

The Security Audit Log	The System Log
Objective	
Records security-related information that can be used to reestablish a series of events (for example, unsuccessful logon attempts or transaction starts).	Records information that may signal system problems (for example, database read errors, rollbacks).
Audience	
Auditors	System administrators
Flexibility of Use	
You can activate and deactivate the Security Audit Log as necessary. Although you may wish to audit your system on a daily basis, you do not have to. For example, you may wish to activate the Security Audit Log for a period of time before a pre-designated audit and deactivate it between audits.	The system log is needed on a continuous basis. You do not deactivate the system log.
Log Availability	
The audit logs are local logs maintained on each application server. However, in contrast to the system log, the system maintains its audit logs on a daily basis and you have to archive or delete the log files manually. This way, you can refer to logs from previous days and the time frame for available logs is increased.	<p>There are two types of system logs, local and central. Local logs are maintained on each individual application server. These files are circular, meaning that once they are full, they are overwritten from the beginning. The sizes of these logs, as well as the time frame where they are available, are limited.</p> <p>You also have the option to maintain a central log. However, the central log is currently not completely platform independent. At the current time, it can only be maintained on a UNIX platform. The central log is also not kept indefinitely.</p>

Comparing the Security Audit Log and the System Log

Handling Sensitive Data

The Security Audit Log contains personal information that may fall under data protection regulations.

The system log does not contain any personal data.

You need to pay close attention to the data protection regulations before you activate the Security Audit Log.

Maintaining Static Profiles

Maintaining Static Profiles

Use

You specify the information you want to audit in filters that you can either:

1. Create and save permanently in the database in static profiles.

If you use this option, all of the application servers use identical filters for determining which events should be recorded in the audit log. You only have to define filters once for all application servers.

You can also define several different profiles that you can alternatively activate.

2. Change dynamically on one or more application servers.

With this option, you can dynamically change the filters used for selecting events to audit. The system distributes these changes to all active application servers.

This topic concentrates on permanently saving filters in static profiles in the database. For information on changing the filters dynamically, see [Changing Filters Dynamically \[Page 242\]](#).



Filters saved in static profiles take effect at the next application server start.

Prerequisites

The following profile parameters must be set:

Audit Log Profile Parameters

Profile Parameter	Description
rsau/enable	Enable the Security Audit Log
rsau/local/file	Names and locations of the audit files
rsau/max_diskspace/local	Maximum space to allocate for the audit files
rsau/selection_slots	Number of filters to allow for the Security Audit Log

Procedure

1. To access the Security Audit Log configuration screen from the *SAP standard menu*, choose *Tools → Administration → Monitor → Security Audit Log → Configuration*.

The *Security Audit: Administer Audit Profile* screen appears with the *Static configuration* tabstrip activated. If an active profile already exists, it is displayed in the *Active profile* field.

2. Enter the name of the profile to maintain in the *Displayed profile* field.
3. If you are creating a new audit profile, choose *Profile → Create*. To change an existing profile, choose *Profile → Display <-> Change*.



To display an existing profile before changing it, choose *Profile → Display*.

Maintaining Static Profiles

The lower section of the screen contains tabstrips for defining filters. The number of tabstrips correspond to the value of the profile parameter `rsau/selection_slots`. Within each tabstrip, you define a single filter.

4. [Define filters \[Page 244\]](#) for your profile.
5. Make sure the *Filter active* indicator is set for each of the filters you want to apply to your audit.
6. Save the data.
7. To activate the profile, choose *Profile → Activate*.
8. Shut down and restart the application server to make the changes effective.

Result

The filters you define are saved in the audit profile. If you activate the profile and restart the application server, actions that match any of the active filter events are then recorded in the Security Audit Log.



On some UNIX platforms, you also need to clear shared memory by explicitly executing the program `cleanipc`. Otherwise, the old configuration remains in shared memory and the changes to the static profile do not take effect.

Changing Filters Dynamically

Changing Filters Dynamically

Use

You specify the information you want to audit in filters that you can either:

1. Create and save permanently in the database in static profiles.

If you use this option, all of the application servers use identical filters for determining which events should be recorded in the audit log. You only have to define filters once for all application servers.

You can also define several different profiles that you can alternatively activate.

2. Change dynamically on one or more application servers.

With this option, you can dynamically change the filters used for selecting events to audit. The system distributes these changes to all active application servers.

This topic concentrates on dynamically changing filters. For information on defining filters in static profiles, see [Maintaining Static Profiles \[Page 240\]](#).



These changes are active until they are changed or the application server is shut down.

Prerequisites

The following profile parameters must be set:

Audit Log Profile Parameters

Profile Parameter	Description
rsau/enable	Enable the Security Audit Log
rsau/local/file	Names and locations of audit files
rsau/max_diskspace/local	Maximum space to allocate for the audit files
rsau/selection_slots	Number of filters to allow for the Security Audit Log

Procedure

1. To access the Security Audit Log configuration screen from the *SAP standard menu*, choose *Tools* → *Administration* → *Monitor* → *Security Audit Log* → *Configuration*.



The *Security Audit: Administer Audit Profile* screen appears with the *Static configuration* tabstrip activated.

2. Choose the *Dynamic configuration* tabstrip or *Goto* → *Dynamic configuration* from the menu.

In the upper section of the screen, you receive a list of the active instances and their auditing status. The lower section of the screen contains tabstrips for maintaining filters.

3. Choose *Configuration* → *Display <-> Change*.
4. [Define filters \[Page 244\]](#) for the application server.

Changing Filters Dynamically

5. Make sure the *Filter active* indicator is set for each of the filters you want to apply to the audit on the application server.
6. If you want to distribute the filter definition to all of the application servers, choose *Configuration → Distribute configuration*.
7. To change the auditing status on a single application server, select the status indicator in the *List of active instances* table.
 -  indicates an activated audit.
 -  indicates a deactivated audit.
8. To activate the filter (or filters) on all of the application servers, choose *Configuration → Activate audit*. (To deactivate the filters on all of the application servers, choose *Configuration → Deactivate audit*.)



If you receive a program failure, then make sure you have the authorization S_RFC with the value SECU in your authorization profile. (The system uses remote function calls to obtain a list of servers and therefore, you need the appropriate authorizations.)

Result

The audit filters are dynamically created on all active application servers. If you activate the profile(s), then any actions that match any of these filters are recorded in the Security Audit Log. Changes to the filter definitions are effective immediately and exist until the application server is shut down.

Defining Filters

Defining Filters

Use

You define the events that the Security Audit Log should record in filters.

You can specify the following information in the filters:

- User
- SAP System client
- Audit class (for example, dialog logon attempts or changes to user master records)
- Weight of event (for example, critical or important)

For examples of filters, see [Example Filters \[Page 255\]](#).

You can define filters that you save in static profiles in the database (see [Maintaining Static Profiles \[Page 240\]](#)) or you can define them dynamically for one or more application servers (see [Changing Filters Dynamically \[Page 242\]](#)).

Prerequisites

- The number of filters you can specify is defined in the profile parameter `rsau/selection_slots`.
- You are either [defining static profiles \[Page 240\]](#) or [changing filters dynamically \[Page 242\]](#) using the Security Audit Log configuration tool. For each allocated filter, a tabstrip appears in the lower section of the screen.

Procedure

1. Select the tabstrip for the filter you want to define.
2. Enter the *Client* and *User* names in the corresponding fields.



You can use the wildcard (*) value to define the filter for all clients or users. However, a partially generic entry such as 0* or ABC* is **not** possible.

3. Select the corresponding *Audit classes* for the events you want to audit.
4. Audit events are divided into three categories, critical, important, and non-critical. Select the corresponding categories to audit.
 - *Only critical*
 - *Important and critical*
 - *All*
5. If you want to define the events to audit more specifically:
 - a. Choose *Detailed configuration*.

A table appears containing a detailed list of the audit classes with their corresponding event classes (critical, severe, non-critical) and message texts. (The message texts correspond to the system log messages AU<X>.)

- b. Select the events you want to audit. You can either:
- Select a single event by activating the *Recording* indicator for a specific event.
 - Select all events for an entire audit class by choosing the audit class descriptor (for example, *Dialog logon*).

- c. Choose *Accept changes*. ✓

The filter tabstrips reappear.



If you have made detailed settings, then the audit class and event class indicators no longer appear in the corresponding filter tabstrip. To cancel the detailed settings and reload the default configuration, choose *Reset*.

6. To activate the filter, select the *Filter active* indicator.
7. Continue with [defining static profiles \[Page 240\]](#) or [changing filters dynamically \[Page 242\]](#).

Displaying the Audit Analysis Report

Displaying the Audit Analysis Report

Use

The Security Audit Log produces an audit analysis report that contains the audited activities. By using the audit analysis report you can analyze events that have occurred and have been recorded on a local server, a remote server, or all of the servers in the SAP System.

The audit analysis report produced by the Security Audit Log is designed analog to the [System Log \[Page 44\]](#).

Procedure

1. To access the Security Audit Log analysis screen from the *SAP standard menu*, choose *Tools → Administration → Monitor → Security Audit Log → Analysis*.

The *Security Audit Log: Local Analysis* screen appears; local analysis is the default.

2. If you want to analyze a remote server, choose *Security Audit Log → Choose → Remote Audit Log*. To analyze all servers, choose *Security Audit Log → Choose → All audit logs*.

Your choice is displayed next to the *Imported audit log* entries field.

3. If you choose to analyze a remote server, then enter the name of the application server in the *Instance name* field.
4. Enter any restrictions you want to apply to the audit analysis report in the appropriate fields or by selecting the desired indicators (for example, *From date/time*, *To date/time*, *User*, *Transaction*, *Audit classes*, or *Events to select*).



Events are classified into three categories, critical, important, and non-critical, with critical being the most important. You can view critical events only, critical and severe events, or all events.

5. If you want to include or exclude specific messages from your report:
 - a. Choose *Edit → Expert mode*.
 - b. Choose *Message filter*.
 - c. Select either *Only these messages* or *All except these messages* as appropriate.
 - d. Enter the message numbers you want to include or exclude. (The message numbers correspond to the system log messages AU<X>.)
 - e. Choose *Use*.
6. To modify the output format, change the options in the *Format* section. For more information, see [The Audit Log Display Options \[Page 254\]](#).
7. To read the Security Audit Log, choose one of the following options:
 - Choose *Security Audit Log → Re-read audit log* to initially read or to replace a previously read log.
 - Choose *Security Audit Log → Re-display only* to view the last audit log you read. For example, you can change the *Selection* options to modify the audit analysis report without having to re-read the log.

Displaying the Audit Analysis Report

- Choose *Security Audit Log* → *Read audit log* to merge new information using different selection criteria with the current information in the audit analysis report.



The *Imported audit log entries* field tells how many log entries the system has read from the log file. When you first enter the *Audit Log: Analysis* initial screen, this field is set to the value "0".

Sorting the Audit Log Display

To sort the audit analysis report, choose *Security Audit log* → *Sort* → <sort option>.

The following sort options are available:

- Write sequence
- Time
- Instance

Result

The result is the audit analysis report containing the messages that correspond to your selection criteria. By selecting an individual message, you can view more detailed information (see [Reading the Audit Analysis Report \[Page 248\]](#)).

Reading the Audit Analysis Report

Reading the Audit Analysis Report

In this section, we describe how to read the audit analysis report produced from the procedure [Displaying the Audit Analysis Report \[Page 246\]](#).

The Main Audit Analysis Report

The audit analysis report is divided into four main sections:

- Introductory information
- Audit report
- Statistical analysis
- Contents

Introductory Information

At the top of the report, you find the selection options applied to the audit file to generate this report (for example, *From date/time*, *To date/time*, *User*, and *Audit classes*).

Audit report

The audit report follows the introductory data and contains the following information for each audit event found in audit file that applies to your selection criteria (depending on your display configuration):

- Date
- Time
- Instance
- Category (dialog or batch)
- Message number
- Audit class code (For example, a dialog logon attempt belongs to class number 002.)
- User
- Transaction code
- Terminal number

Summary information is included at the end of the list (for example, the number of records read, the number of records selected, and audit file names).

Statistical analysis

If you included *With statistical analysis* in the display options, then the following blocks of information are included after the audit data:

- Instance statistics (when analyzing all instances)
- Client statistics
- Report statistics
- Transaction statistics

Reading the Audit Analysis Report

- User statistics
- Message statistics

Contents

A list of contents is provided at the end of the report.

The Detailed Audit Analysis Report

To view details about a specific message, place the cursor on the entry and choose *Edit* → *Details*. A detailed description of the message including information such as the task name, class, message documentation, and the technical details of the audit record appears.

Deleting Old Audit Files

Deleting Old Audit Files

Use

The Security Audit Log saves its audits to a corresponding audit file on a daily basis. Depending on the size of your SAP System and the filters specified, you may be faced with an enormous quantity of data within a short period of time.



We recommend archiving your audit files on a regular basis and deleting the original files as necessary.

Use this procedure to delete old audit files. You can either delete the files from all application servers or from only the local server where you are working. If an application server is not currently active, it will be included in the next reorganization.



This procedure only deletes the audit log file(s)! It does **not** perform any other administrative tasks such as archiving. If archives are necessary for future references, you must manually archive them before deleting.



You cannot purge files that are less than 3 days old!

Procedure

1. To access the Security Audit Log reorganization tool from the *SAP standard menu*, choose *Tools → Administration → Monitor → Security Audit Log → Reorganization*.

The *Security Audit: Delete Old Audit Logs* screen appears.

2. Enter the *Minimum age* of files to delete (default = 30 days).

This value must be > 3.

3. Activate the *To all active instances* indicator to delete the audit files from all application servers. Leave the indicator blank if you only want to delete the files from the local application server.
4. Activate the *Simulation only* indicator if you do not actually want to delete the files. In this case, the action is only simulated.
5. Choose *Audit Log → Continue*.

Result

The system deletes the corresponding audit files (unless you chose to simulate). You receive a list showing how many files were deleted and how many were retained on each application server.

Security Alerts in the CCMS Alert Monitor

When the Security Audit Log records events, it also triggers a corresponding security alert in the Computing Center Management System (CCMS) alert monitor.

The security alerts that are created correspond to the audit classes of events as defined in the Security Audit Log, which include:

- Dialog logon attempts
- RFC/CPIC logon attempts
- Transaction starts
- Report starts
- RFC function calls
- Changes to user master records
- Changes to the Security Audit Log configuration

By monitoring the security alerts in the CCMS alert monitor, you can quickly identify security-related problems in your system. After performing the immediate on-alert action to resolve the alert, you can analyze the Security Audit Log files for more information about the specific event that caused the alert.

You can view the security alerts directly in the CCMS alert monitor (see the topic [Viewing Security Alerts \[Page 252\]](#)) or use BAPIs (Business Application Program Interfaces) to access the alerts from external programs (see the topic [Reading Security Alerts Using BAPIs \[Page 253\]](#)).

Viewing Security Alerts

Viewing Security Alerts

Prerequisites

The Security Audit Log must be activated on the application server so that the event is also triggered in the CCMS alert monitor.

Procedure

1. To access the CCMS alert monitor from the *SAP standard menu*, choose *Tools → CCMS → Control/Monitoring → Alert monitor*.

The *CCMS monitor sets* appear.

2. To locate the security alerts, expand the node *SAP CCMS Monitor Templates*.
3. Place the cursor on the *Security* node and choose *Monitor → Load monitor*.

The *Security* monitor appears.

The alerts triggered by the Security Audit Log are located under the nodes for each application server.

4. Expand the node for the specific application server (or servers) that you want to examine.

The categories that appear correspond to the audit classes recorded in the Security Audit Log. Entries with active alerts are indicated in red or yellow, depending on the highest alert level (critical or important) existing in the category.

5. Select the categories you want to examine on each application server or the complete application server node.
6. Choose *Edit → Alerts → Display alerts*.

A list containing the chosen categories appears.

7. Process the alerts as necessary.



For more information on the CCMS alert monitor and how to process alerts, see [The Alert Monitor \[Ext.\]](#).

Reading Security Alerts Using BAPIs

The security alerts are also available to external programs using BAPIs (Business Application Programming Interfaces). The report RSAU_READ_AUDITLOG_EXTERNAL is a sample SAP program that you can use as a template for accessing the security alerts using BAPIs.

The Audit Log Display Options

The Audit Log Display Options

Option	Meaning
<i>No. pages for individual entries</i>	Specifies the maximum number of pages you want to view. This only applies to the main section of the report, not to the introductory information or summaries.
<i>With statistical analysis</i>	<p>If you activate this option, then the following statistics are included with your report.</p> <ul style="list-style-type: none">• Instance statistics (when analyzing all instances)• Client statistics• Report statistics• Transaction statistics• User statistics• Message statistics
<i>Settings</i>	Specifies the layout and output devices.

Example Filters

In the following example, the Security Audit Log is enabled on the server `pawdf050`. The active profile is `PROFILE1`.

For *Filter 1*, the system will record any dialog logon attempts, RFC or CPIC logon attempts, or transaction starts that are categorized as important or critical events. The events are recorded for all users and for events in any of the SAP System clients.

For *Filter 2*, the system only records events categorized as critical in client 000 for `TESTUSER`.

Both filters are active in the system.

Filter 1

Example Filters

The screenshot shows the 'Security Audit: Change Audit Profile' dialog box in the SAP GUI. The window has a menu bar with 'Profile', 'Edit', 'Goto', 'Environment', 'System', and 'Help'. Below the menu is a toolbar with various icons. The title bar reads 'Security Audit: Change Audit Profile'. The main area is divided into two tabs: 'Static configuration' and 'Filter 1' (selected). Under 'Static configuration', there are two input fields: 'Active profile' and 'Displayed profile', both containing the text 'PROFILE1'. Below this, there are two sub-tabs: 'Filter 1' and 'Filter 2'. Under 'Filter 1', there is a checkbox 'Filter active' which is checked. To its right are 'Reset' and 'Detail configurat' buttons. Below these are three columns: 'Selection criteria', 'Audit classes', and 'Events'. The 'Selection criteria' column has two rows: 'Client' with a yellow square icon and a lock icon, and 'User names' with an asterisk. The 'Audit classes' column has a list of checkboxes: 'Dialog logon' (checked), 'RFC/CPIC logon' (checked), 'RFC function call' (unchecked), 'Transaction start' (checked), 'Report start' (unchecked), 'User master change' (unchecked), and 'Other' (unchecked). The 'Events' column has three radio buttons: 'Only critical' (unchecked), 'Important and critical' (checked), and 'All' (unchecked). At the bottom of the dialog, there is a status bar with the text 'A9B (1) (000)' and 'pawdf050 INS'.

Profile Edit Goto Environment System Help

Security Audit: Change Audit Profile

Static configuration

Active profile PROFILE1

Displayed profile PROFILE1

Filter 1 Filter 2

☒ Filter active Reset Detail configurat

Selection criteria	Audit classes	Events
Client	<input checked="" type="checkbox"/> Dialog logon	<input type="radio"/> Only critical
User names *	<input checked="" type="checkbox"/> RFC/CPIC logon	<input checked="" type="radio"/> Important and critical
	<input type="checkbox"/> RFC function call	<input type="radio"/> All
	<input checked="" type="checkbox"/> Transaction start	
	<input type="checkbox"/> Report start	
	<input type="checkbox"/> User master change	
	<input type="checkbox"/> Other	

A9B (1) (000) pawdf050 INS

Filter 2

The screenshot shows the SAP Security Audit: Change Audit Profile dialog box, specifically the Filter 2 tab. The dialog has a menu bar (Profile, Edit, Goto, Environment, System, Help) and a toolbar. The title bar reads "Security Audit: Change Audit Profile".

Static configuration

Active profile: PROFILE1
Displayed profile: PROFILE1

Filter 1 | Filter 2

☒ Filter active

Selection criteria	Audit classes	Events
Client: 000	<input checked="" type="checkbox"/> Dialog logon	<input checked="" type="radio"/> Only critical
User names: TESTUSER	<input checked="" type="checkbox"/> RFC/CPIC logon	<input type="radio"/> Important and critical
	<input checked="" type="checkbox"/> RFC function call	<input type="radio"/> All
	<input checked="" type="checkbox"/> Transaction start	
	<input checked="" type="checkbox"/> Report start	
	<input checked="" type="checkbox"/> User master change	
	<input checked="" type="checkbox"/> Other	

At the bottom, the status bar shows: A9B (1) (000) | pawdf050 | INS

Example Filters