

# The SAP Lock Concept (BC-CST-EQ)



HELP.BCCSTEQ

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

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

ORACLE<sup>®</sup> is a registered trademark of ORACLE Corporation.

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

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

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

JAVA<sup>®</sup> is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT<sup>®</sup> 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

## Inhalt

<b>The SAP Lock Concept (BC-CST-EQ)</b> .....	<b>5</b>
Functions of the R/3 Lock Concept .....	6
The Owner Concept .....	10
The Lock Table .....	13
Lock Collisions .....	16
Cumulation of Locks .....	19
Questions and Answers Regarding Locks .....	21
Important Profile Parameters for the Lock Concept.....	25
Managing Lock Entries .....	27
Choosing and Displaying Lock Entries .....	29
Deleting Lock Entries.....	32
Testing Lock Management .....	34
Displaying Lock Statistics .....	35
Analyzing and Rectifying Problems .....	37

## 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 \[Extern\]](#). A description of handling lock objects is provided in the ABAP Dictionary Documentation under [Lock Objects \[Extern\]](#).

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 \[Extern\]](#).

### Features

You can use the [Lock Management \[Seite 27\]](#) 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 \[Seite 6\]](#).

The most important profile parameters for the R/3 lock concept are described [here \[Seite 25\]](#). You can use these parameters to tailor your system resources to your needs.

## Functions of the R/3 Lock Concept

# 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 \[Seite 27\]](#).

## 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 \[Extern\]](#) 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 \[Extern\]](#) 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 \[Seite 13\]](#). This server runs on the central instance.

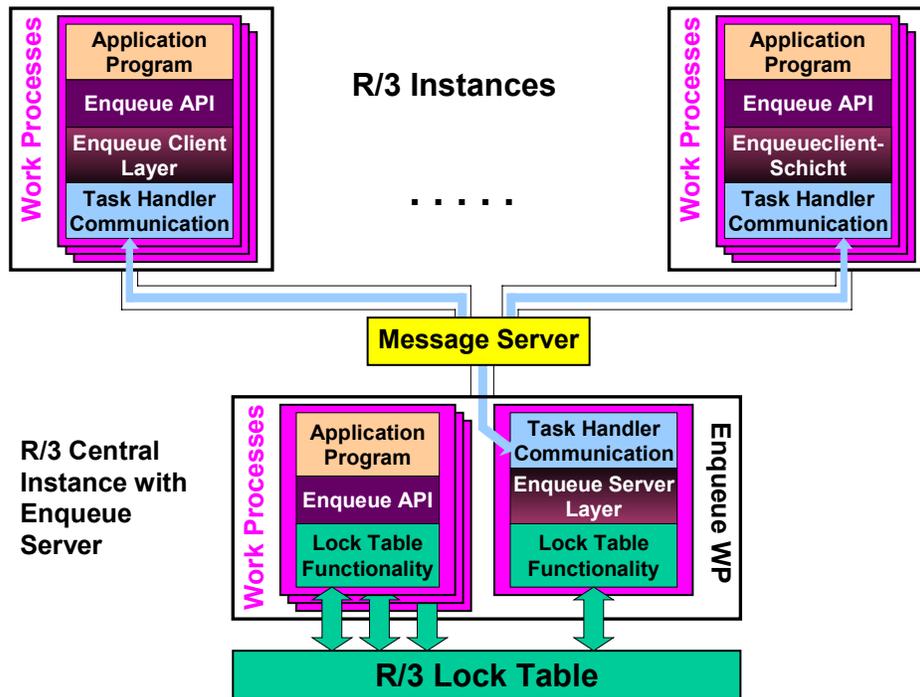
If a lock is to be set in an application running on an instance other than the central instance (for example, on a different host), the lock request is transferred via the [dispatcher \[Extern\]](#) and message server to the dispatcher of the central instance, which then forwards it to the enqueue work process. This process then checks the lock table to determine whether the lock request collides with an existing lock (see also [Lock Collisions \[Seite 16\]](#)). 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.

### Enqueue communication



### Locks and R/3 Update

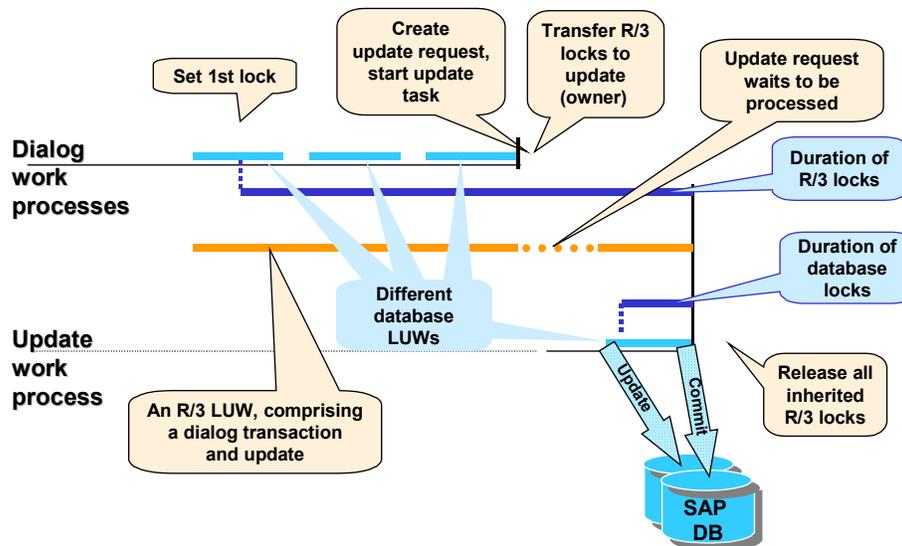
During the course of the transaction, the lock is transferred to the [update in R/3 \[Extern\]](#). This procedure is described in detail under [The Owner Concept \[Seite 10\]](#) and [Function Modules for Lock Requests \[Extern\]](#). Locks that have been transferred to the update are stored both in 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 \[Extern\]](#).

## Functions of the R/3 Lock Concept

## 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 \[Extern\]](#)). 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 \[Extern\]](#)).

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

[The Owner Concept \[Seite 10\]](#)

[The Lock Table \[Seite 13\]](#)

[Lock Collisions \[Seite 16\]](#)

[Cumulation of Locks \[Seite 19\]](#)

[Questions and Answers Regarding Locks \[Seite 21\]](#)

## The Owner Concept

## The Owner Concept

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

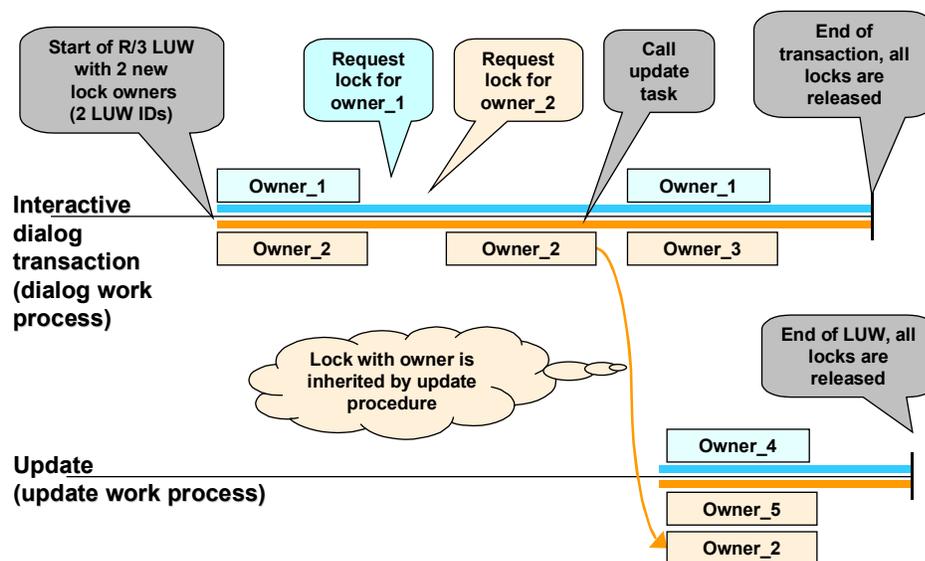
An owner is identified by his or her owner ID, as described in the section entitled [The Lock Table \[Seite 13\]](#).

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 \[Extern\]](#)).

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



## SAP LUW: Dialog Transaction with Update



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 \[Extern\]](#)), 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.

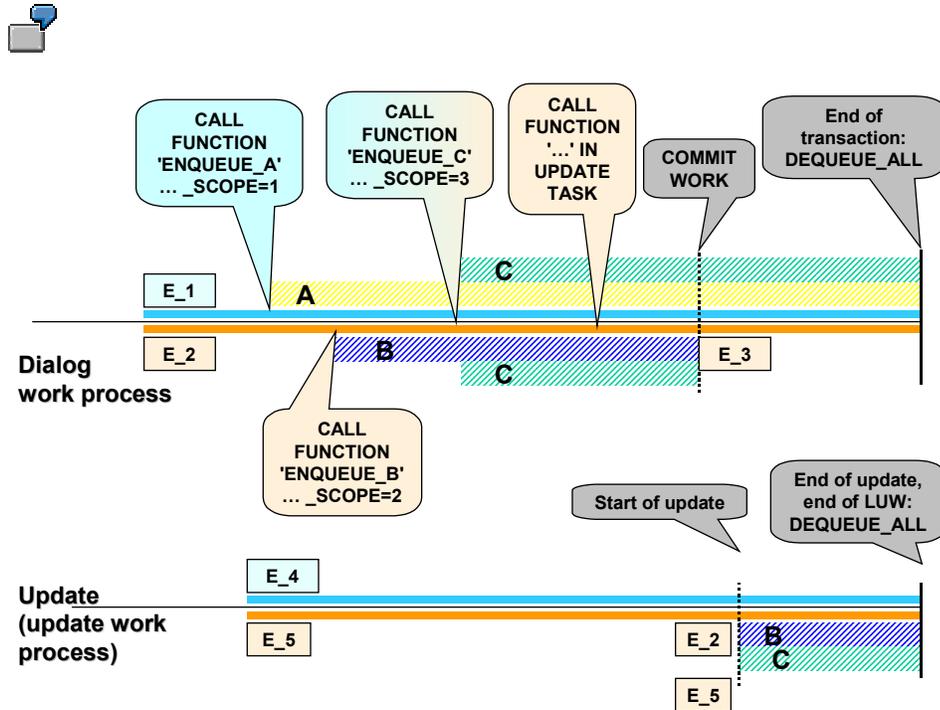
The following diagram shows the locks during the course of an SAP LUW in conjunction with the `_SCOPE` parameter. The application programmer uses this parameter to determine which of the two owners should actually own the lock. The diagram also shows how long the R/3 locks are active (these are not the actual database locks, as described under [Functions of the R/3 Lock Concept \[Seite 6\]](#)).

`_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 \[Extern\]](#) 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 \[Extern\]](#). 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 Owner Concept**

[The Lock Table \[Seite 13\]](#)

[Lock Collisions \[Seite 16\]](#)

[Cumulation of Locks \[Seite 19\]](#)

## 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 \[Seite 16\]](#)). 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

Owner_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.	
Owner_2	The above applies here to owner_2	
Backup Id	Backup Id (index indicating where the lock entry is stored in the backup file) and backup flag (0 (no backup) or 1 (backup)).	
Elementary lock	Lock mode	<b>S</b> (Shared lock, read lock) <b>E</b> (Exclusive lock, write lock) <b>X</b> (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 \[Seite 27\]](#).



Lock Collisions

## 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 \[Seite 13\]](#). 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 \[Seite 19\]](#).)

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 elementary 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 5<sup>th</sup> 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 \[Seite 10\]](#)).

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 \[Extern\]](#))

The following graphic shows examples of these situations, where O\_x is an owner different to O\_1 and O\_2.



Lock Collisions

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 \[Seite 19\]](#)

[The Owner Concept \[Seite 10\]](#)

## Cumulation of Locks

As described under [Lock Mode \[Extern\]](#), 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 \[Seite 10\]](#)) 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.

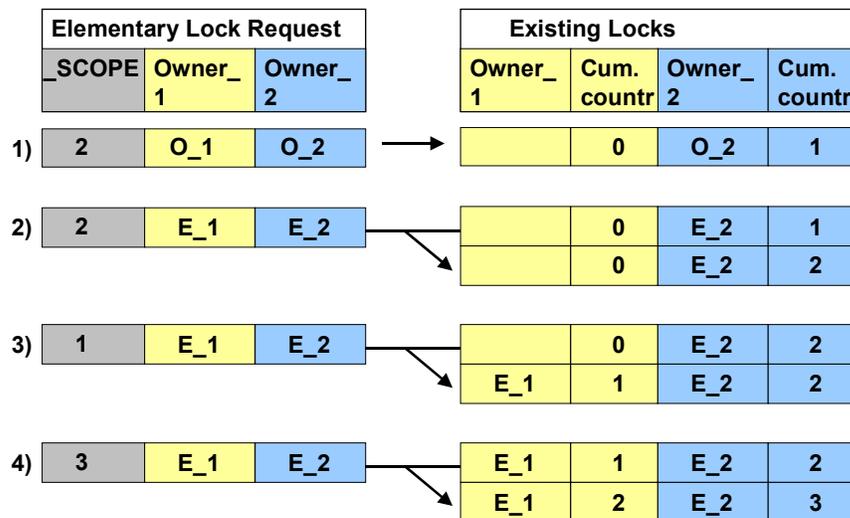
The type of lock selected is determined by the programmer of the transaction when the lock object is created. See also [The R/3 Lock Concept \[Extern\]](#) in the ABAP Documentation.

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 \[Seite 13\]](#). 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

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 \[Seite 6\]](#).

### 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 \[Seite 13\]](#).

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 I 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 \[Extern\]](#) in the CCMS documentation.

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

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 <a href="#">lock table [Seite 13]</a> 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 <a href="#">System Logs [Extern]</a>).</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 <a href="#">displaying the lock statistics [Seite 35]</a>.</p> <p><b>Unit:</b> kilobyte <b>Default value:</b> 4096</p>
rdisp/wp_no_enq	<p>Number of enqueue work processes that are to run on this instance.</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><b>Unit:</b> integer <b>Default value:</b> 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 <a href="#">Functions of the R/3 Lock Concept [Seite 6]</a>).</p> <p><b>Unit:</b> character string (path name)</p> <p><b>Default value:</b> /usr/sap/&lt;SID&gt;/D&lt;instance no.&gt;/log/ENQBCK</p>
rdisp/enqname	<p>Name of the application server that provides the enqueue service (see <a href="#">Functions of the R/3 Lock Concept [Seite 6]</a>).</p> <p>The name should be maintained in the default profile so that all application servers use the same name.</p> <p><b>Unit:</b> character string (instance)</p> <p><b>Default value:</b> 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 \[Seite 29\]](#)).

This enables you to detect and rectify problems, for example, by [deleting \[Seite 32\]](#) 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 \[Extern\]](#).

### Features

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

The screenshot shows the SAP R/3 interface for selecting lock entries. The window title is "SAP R/3". The menu bar includes "Sperreintrag", "Bearbeiten", "Springen", "Zusätze", "System", and "Hilfe". The toolbar contains various icons for navigation and actions. The main area is titled "Sperreinträge selektieren" and features an "Auflisten" button. Below this, there are input fields for "Tabellen-Name", "Sperr-Argument", "Mandant" (set to "000"), and "Benutzer-Name" (set to "MUSTER"). At the bottom right, there is a status bar showing "BIN (1) (000)", "is0201", and "INS".

You can use the lock management facility to

- [Choose and display lock entries \[Seite 29\]](#)
- [Delete lock entries \[Seite 32\]](#)
- [Test the lock management facility \[Seite 34\]](#)
- [Display the lock statistics \[Seite 35\]](#)

## Managing Lock Entries

### Activities

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

## Choosing and Displaying Lock Entries

You can display lock entries according to the following criteria (see also [Functions of the R/3 Lock Concept \[Seite 6\]](#)):

**Sperreinträge selektieren**

Auflisten

Tabellen-Name	
Sperr-Argument	
Mandant	000
Benutzer-Name	MUSTERMANN

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		SCRP_ENQ	1300SAPMSLIC
000	BAREISST	14:56:17		TRDIR	VERDATA0
000	FANH	14:59:43		SCRP_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 \[Extern\]](#) 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 \[Seite 13\]](#).

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 \[Seite 10\]](#)), 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

## 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 \[Seite 21\]](#)

### 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 \[Extern\]](#).)

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 \[Extern\]](#)) 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 \[Extern\]](#).

- 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**

## 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 \[Seite 6\]](#).
- *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 \[Seite 21\]](#) 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

### 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 ( <b>DEQUEUE</b> )
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

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