MQSeries link for R/3

User's Guide

Version 1.0

GC33-1934-00

MQSeries link for R/3

User's Guide

Version 1.0

GC33-1934-00

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix D, “Notices” on page 93.

First Edition (January 1997)

This edition applies to:

IBM MQSeries link for R/3 Version 1.0 for AIX, program number 5765-B66
IBM MQSeries link for R/3 Version 1.0 for HP-UX, program number 5765-C01
IBM MQSeries link for R/3 Version 1.0 for Sun Solaris, program number 5765-B98
IBM MQSeries link for R/3 Version 1.0 for Windows NT, program number 5765-B99

and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page titled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Contents

Figures

Figures and tables

About this book

The *MQSeries link for R/3 User's Guide* describes the MQSeries link for R/3 product: what it is, how to install it on your system, and how you can use it.

This book includes these chapters:

Who this book is for

This User's Guide is written for anyone who runs business applications that use R/3 and who needs to implement commercial messaging using the MQSeries family of products. This book will be of particular interest if you intend to design, program, implement, administer, maintain, or support systems that use the MQSeries link for R/3.

To use this book, you should have a general understanding of R/3 application systems together with some experience or knowledge of messaging using the MQSeries family of products.

Where to find more information about MQSeries

There are many MQSeries publications to help you use the SAP link. You may find the following books particularly useful:

- *MQSeries An Introduction to Messaging and Queuing, GC33-0805*
- *MQSeries Message Queue Interface Technical Reference, SC33-0850*
- *MQSeries Planning Guide, GC33-1349*
- *MQSeries Command Reference, SC33-1369*
- *MQSeries Clients, SC33-1632*
- *MQSeries Application Programming Reference, SC33-1673*
- *MQSeries Application Programming Reference Summary, SX33-6095*
- *MQSeries Application Programming Guide, SC33-0807*
- *MQSeries Distributed Queuing Guide, SC33-1139*

About this book

In addition, there are *MQSeries System Management Guides* for all the platforms supported by MQSeries. Each of these publications includes a complete list of the available MQSeries publications.

Information about MQSeries on the Internet

The URL of the MQSeries home page on the Internet is:

```
http://www.hursley.ibm.com/mqseries/
```

Where to find more information about R/3

You can find a description of the Application Link Enabling (ALE) distribution strategy in the *ALE Consultants Manual*, available from SAP.

Introduction to MQSeries link for R/3

This chapter gives you an overview of the MQSeries link for R/3, referred to in this book as *SAP link*. It gives you a general description of the product, including the SAP link servers, and the user exits.

Overview

MQSeries link for R/3 (SAP link) is an interface that enables you to integrate your R/3 applications with applications running in other environments, including those on SAP R/3 (referred to in this book as the R/3 system) and R/2 systems.

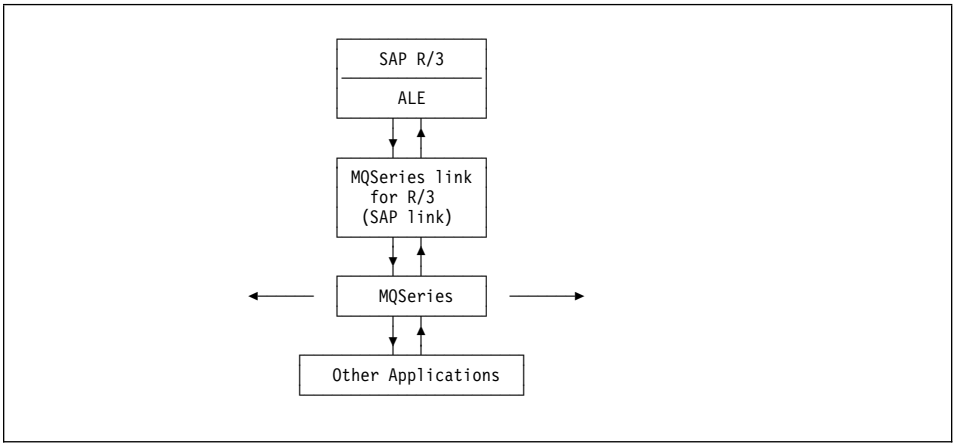


Figure 1. MQSeries link for R/3. SAP link connects R/3 systems to other applications that use MQSeries messaging.

SAP link works with the Application Link Enabling (ALE) layer of the R/3 system to transmit Intermediate Documents (IDocs) into and out of your R/3 system, using MQSeries messages and queues to carry the information. It extends the scope of your business by allowing you to link your R/3 applications to any other application that you can access through MQSeries, even when those applications require different data formats.

In summary, you can use MQSeries link for R/3 to connect an R/3 system to:

- Other R/3 systems.
- R/2 systems.
- Any other system that you want to exchange data with an R/3 system, such as a legacy database holding enterprise data that you need to get into R/3.

For more information about converting data between different types of systems, see “Understanding user exits” on page 7.

Introduction

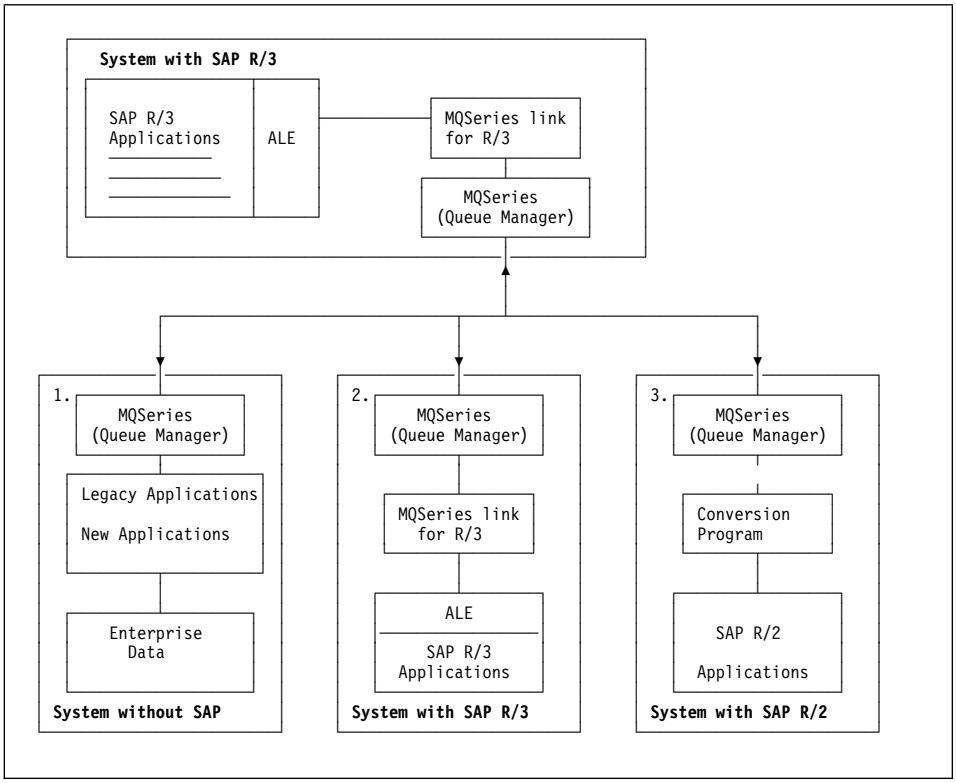


Figure 2. SAP link overview. SAP link enables applications running under R/3 to exchange information with (1) Legacy (or new), (2) Other R/3, or (3) SAP R/2 applications.

The benefits of using SAP link

SAP link provides robust, reliable middleware that connects SAP and other programs across many IBM and non-IBM platforms. It uses programming resources efficiently, makes applications adaptable, and eases network management. It gives programs independence from communications protocols and from the non-availability of the systems, networks, and programs.

With SAP link you can link your existing R/3 business applications with your other applications, including applications running under older versions of SAP, such as R/2, and non-SAP legacy systems.

Understanding SAP link servers

There are two main functional components associated with the SAP link product. These are known as the *outbound server* and the *inbound server*. The outbound server is used when sending data from an R/3 system; the inbound server is used when receiving information into R/3. Both servers have *exit handlers* to give you access to *user exits* outside SAP link for performing special functions such as data conversion and translation. If you use a user exit, you can either write your own conversion program or use a data translator tool.

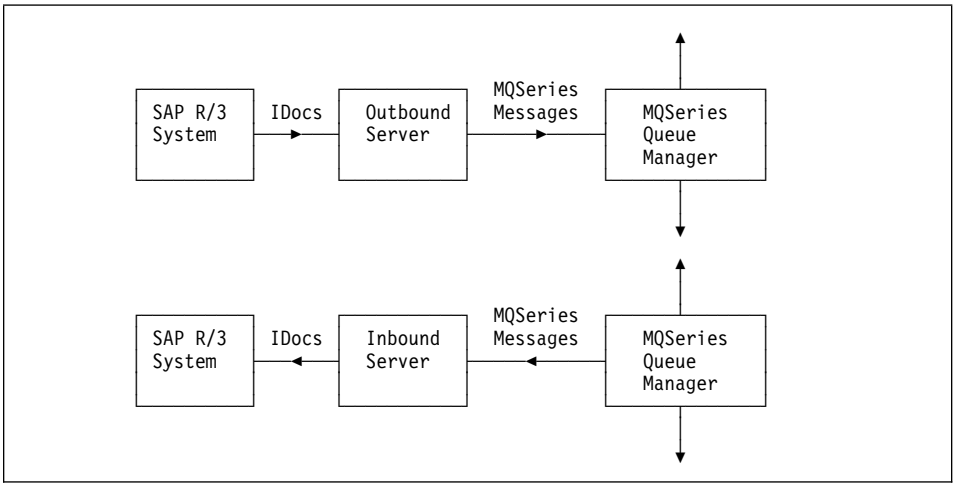


Figure 3. Inbound and outbound servers

Outbound servers

You start the outbound server using the **smqso** command. Thereafter, the server automatically gets control when an R/3 application sends information through MQSeries. The R/3 application passes a single transaction to SAP link as a set of one or more IDocs. The outbound server builds the messages and passes them to MQSeries.

Outbound sequence of events

The following describes the sequence of events when R/3 sends a transaction consisting of one IDoc or multiple IDocs (known as a batch) outbound from an R/3 application (see Figure 4):

1. An R/3 application program creates one or more IDocs representing a single transaction, and uses an asynchronous remote function call (aRFC) to start the outbound process.

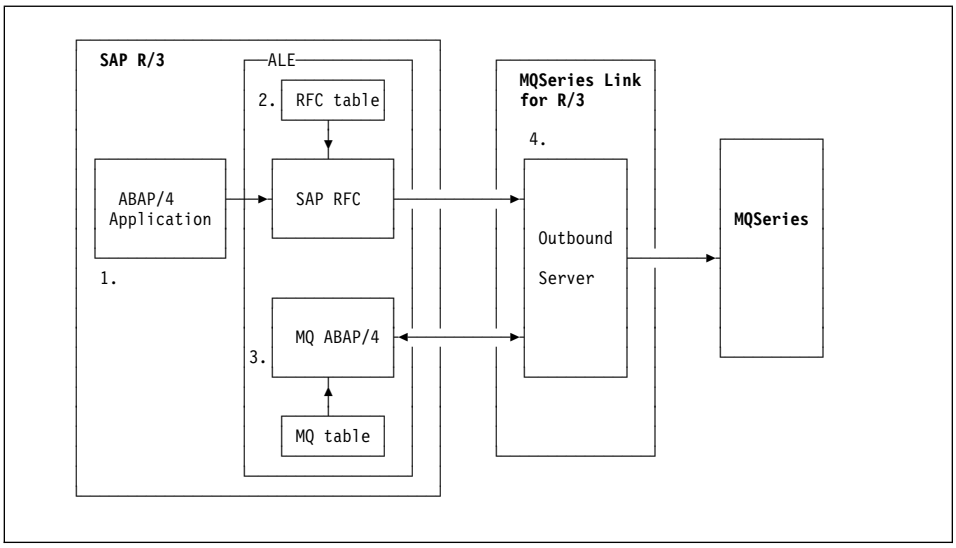


Figure 4. Outbound transactions

2. The RFC component handles the connection to SAP link and provides the additional control information required to handle the transaction and send it to the specified destination.

   The RFC component uses the RFC table (RFCDEST) to identify the connection type and physical destination of the transaction. The table contains entries for all destinations and is maintained using the standard R/3 transaction, `sm59`.

   If the RFC destination program ID matches that of a running outbound server, SAP link is used.

3. SAP link accesses the message queuing (MQ) ABAP/4 function to retrieve additional information needed for an MQSeries destination. The MQ ABAP/4 function uses the destination name as a key to access the MQ table containing the additional information. This includes target queue name, logged on user ID, and whether an outbound user exit should be called. The MQ table is maintained through the SAP link administration panels, accessed through R/3 transaction `sm59`.

4. The RFC component passes the transaction data and control information to the SAP link outbound server. The SAP link outbound server uses MQSeries destination information (from the control information) and the transaction data to construct the MQSeries messages. Each message contains one or more IDocs for a specific destination.

   If the control information indicates that data formatting or conversion is required, the outbound exit handler calls a user exit to perform the translation.

   The outbound server sends the messages to the specified MQSeries queue. All the messages are put under syncpoint; they are treated as a single logical unit of work.

RFC destinations

Within R/3, you must specify the parameters that uniquely identify the external program, SAP link. MQSeries destinations are type T, that is, TCP/IP. Type T destinations refer to external programs that use R/3 RFC libraries to act as RFC servers. In R/3, you must specify a program ID, which you can supply yourself; when you start an SAP link server, make sure that you specify this same program ID on the start server command.

Also, if you are not registering by using the current applications server as your gateway, you must specify the names of an SAP gateway host and service.

SAP provides an R/3 transaction, `sm59`, for specifying these parameters.

Inbound servers

You start the inbound server using the **smqsi** command. The server gets control whenever an MQSeries message is put on the inbound server queue. The inbound server gets the message from the queue and, if it is in the correct format, passes the information in the message as IDocs to the receiving R/3 application.

Note: This means that you have to take steps to ensure that the information is formatted correctly for R/3 processing.

Inbound sequence of events

The following describes the sequence of events when SAP link receives an inbound transaction from MQSeries for an R/3 application.

Introduction

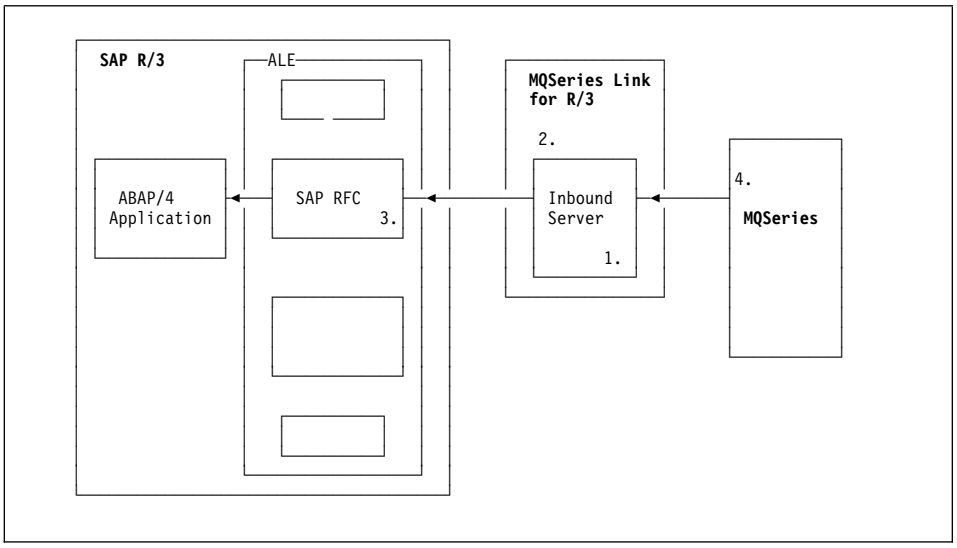


Figure 5. Inbound transactions

1. Once it has been started, the inbound server polls the MQSeries queue for new messages. An application sending information to an R/3 application puts a message on an MQSeries queue. The SAP link inbound server gets the message from the MQSeries queue, under syncpoint. SAP link creates IDocs from the incoming transaction data. If the transaction data is to be formatted or converted in some way, SAP link uses the exit handler to access a user exit written to perform the required processing.
2. The inbound server requests a transaction ID from the R/3 system.

   SAP link keeps track of inbound transactions and manages any messages by making an entry in an internal transaction store queue.
3. The link header in the incoming message data contains information about the R/3 application that is to receive the transaction.

   If you do not specify the information about the remote R/3 system within the R/3 transaction `sm59`, the link header outbound message does not contain all the required destination information. To connect to the remote system, SAP link uses the default values specified in the initialization (.ini) file for the inbound server.

   The RFC program waits for an inbound message from SAP link. When a message arrives, the RFC program executes the INBOUND_IDOC_PROCESS function.
4. After the transaction has been transferred successfully to the R/3 system, SAP link executes an MQSeries commit to delete the inbound message. SAP link also deletes its entry from the internal transaction store queue.

Understanding user exits

The SAP link servers provide user exits, which allow you to process messages as they pass through a server. The exits are called by the exit handler associated with each server. Typically, you use the exits to add extra processing functions that are not directly provided by SAP link or MQSeries, such as reformatting or compressing data in IDocs coming from an R/3 system. You can do these types of operations using your own program or a SAP-approved data translator.

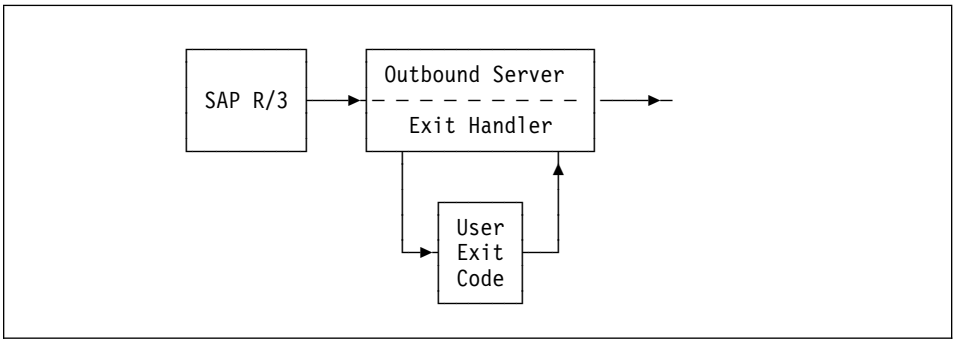


Figure 6. Exit handlers and user exits on the outbound server. You modify the supplied user exit samples to perform any functions that you require.

As with all MQSeries exits, the SAP link exit handlers execute your user-supplied software in-line, without additional message flows. In SAP link there are two forms of user exits; an outbound user exit is called before a message is sent to an MQSeries destination queue, an inbound user exit is called after a message has been retrieved from the inbound server message queue.

An example of when to use exits

An R/3 application sends some invoice data to a non-SAP application. The data from the R/3 application is always in IDoc format. However, the receiving application expects the invoice data in a different format.

To overcome this problem, you can write a user exit for the outbound server to:

- Extract the relevant data from the IDoc in the message
- Build a new message with the data in the format expected by the receiving application

The server then puts the new message on its outbound message queue, where MQSeries applications can access it.

For the inbound server, you can use the reverse procedure to convert incoming data from a non-SAP application into an R/3 IDoc format.

Introduction

Summary of the SAP link components

Here is a brief overview of some of the components and functions that make up the MQSeries link for R/3 product:

Outbound server
The SAP link component that accepts IDocs from the source R/3 system and makes them available to other applications connected by MQSeries.

Inbound server
The SAP link component that receives information from applications connected by MQSeries and passes the information to the destination R/3 system.

R/3 administration panels
The R/3 panels that allow an administrator to configure the SAP link software.

C source header file
A header file containing structure definitions required to process SAP link message data. This is useful for writing your own user exits.

Sample user exit
C source code for a sample user exit that you can use as a basis for your own user exits.

Initialization files
These are text files from which the SAP link server start commands obtain any startup parameters that you do not explicitly specify on the command line.

Sample MQSC files
These are MQSeries command files that create a sample set of MQSeries objects for SAP link.

Planning for MQSeries link for R/3

This chapter tells you about the hardware and software you need to run MQSeries link for R/3 (SAP link).

Hardware requirements

To install and run this product, you need about 5MB of available hard disk space. There are no additional hardware requirements except for those listed for MQSeries and R/3 on the platform you are using. Please consult the installation and planning sections in the appropriate books for these products.

Software requirements

To run SAP link, you must have installed the following:

- At least one R/3 system, version 3.0E or later. You can use version 3.0D, if you apply upgrade BINK090538. This fix will be supplied by SAP AG; details of how to get the fix are supplied with your R/3 system.

  Note: The fix adds MQSeries Options to the Destination Menu if you specify SAP administration function sm59.

- At least one instance of MQSeries Version 2.2.1 for the chosen platform.
- The relevant MQSeries link for R/3 product for the chosen platform.

Restrictions

The maximum size of MQSeries messages (4MB) limits the amount of R/3 IDoc data that can be sent in a single message. Because the SAP link header fields are part of the space that is assigned to MQSeries messages, the actual maximum amount of space available for R/3 transaction data is less than the full 4MB. R/3 transaction data can consist of one IDoc or batch IDocs.

SAP AG recommends a maximum of 2MB for each IDoc.

Planning

Supported platforms

MQSeries link for R/3 is available for the following release levels of these platforms:

- AIX 4.1
- HP-UX** 10.01
- Sun Solaris** 2.5
- Windows NT 3.5.1

MQSeries platforms

Using SAP link, you can connect your R/3 system to other platforms that run MQSeries. Currently, you can run MQSeries on the following platforms:

AIX	AT&T GIS UNIX**
Digital VMS VAX**	HP-UX**
MVS/ESA (CICS and IMS)	OS/2
OS/400	SCO UNIX**
SINIX and DC/OSx**	SunOS**
Sun Solaris**	Tandem NonStop Kernel**
UnixWare**	VSE/ESA
Windows NT	Windows 3.1
Windows 95	

Installing MQSeries link for R/3

This section will tell you how to install MQSeries link for R/3 on the AIX, HP-UX, Sun Solaris, and Windows NT platforms.

It is assumed that MQSeries has already been installed on your platform.

Read me file

Before starting to install SAP link, review any READ.ME file that may be included on the distribution media.

The READ.ME file contains any product and documentation updates that have become available after this book was published.

Installing MQSeries link for R/3 on AIX Version 4.1

1. Go to SMIT with root authority. From shell, enter `smit`
2. Select the device appropriate for your installation using the following sequence of windows:

   ```
   Software Installation & Maintenance
      Install and Update Software
         Install/Update Selectable Software (Custom Install)
            Install Software Products at Latest Available Level
               Install New Software Products at Latest Level
   ```

   Select the CD-ROM drive as the input device:
   /dev/cd0
3. Press `Do` to display parameters for Install Latest Level.
4. Enter `all_licenced` into Software to install, or select the List button to display the multi-select window.

   Set COMMIT Software Updates to NO

   Set SAVE replaced files to YES
5. Press `Do` to install.

For further information on using installp to install software packages, see the AIX documentation.

Note: If you mount the CD-ROM on /cdrom using smit, the product manual is then available in postscript format on the CD-ROM in the directory /cdrom/smq_rsaix41/docs.

Installation

Installing MQSeries link for R/3 on HP-UX Version 10

1. Login to your system as root (or su -).
2. Mount the CD-ROM on /cdrom.
3. Use the HP-UX swinstall program to install the software by typing the following command:

```
swinstall -s /cdrom/smq_hpux10/smq/smq100.img
```

For further information on using swinstall to install software packages, see the HP-UX documentation.

Note: The product manual is available in postscript format on the CD-ROM in the directory /cdrom/smq_hpux10/docs.

Installing MQSeries link for R/3 on Sun Solaris Version 2.5

1. Login to your system as root (or su -).
2. Check to see if the Volume Manager is running on your system by typing the following command:

```
/usr/bin/ps -ef | /bin/grep vold
```

   If it is running, the CD is mounted on `/cdrom/unnamed_cdrom/smq_solaris` automatically. If it is not running, mount the CD by typing the following commands:

```
mkdir -p /cdrom/smq_solaris
mount -F hsfs -r /dev/dsk/cntndnsn /cdrom/smq_solaris
```

   substituting cntndnsn with the name of your CD-ROM device.

3. Use the Solaris pkgadd program to install the software by carrying out the following procedure:
   a. Type `pkgadd -d /cdrom/smq_solaris`

      Note: Add `unnamed_cdrom` to the directory structure if the CD was running already.

   b. Follow the screen prompts.

For further information on using pkgadd to install software packages, see the Solaris documentation.

Note: The product manual is available in postscript format on the CD-ROM in the directory /cdrom/smq_solaris/docs.

Installing MQSeries link for R/3 on Windows NT Version 3.5

1. Login to your systems as administrator user
2. Insert the CD-ROM in the CD-ROM drive, assumed to be e:\
3. Run SETUP.EXE. One method to do this is as follows:
   a. From the Program Manager, select the File menu and click on Run.
   b. A window titled Run is displayed. In the Command Line field, type:

      ```
      e:\smq_nt351\smq\setup.exe
      ```

      Press ENTER or click on OK.
4. Follow the screen prompts.

Note: The product manual is available in postscript format on the CD-ROM in the directory \smq_nt351\docs.

Directories after installation

For each platform, the directory structure that will exist after installation are as follows. In each case there are three new directories and a message catalog.

AIX

```
/usr/lpp/smq/bin
/usr/lpp/smq/include
/usr/lpp/smq/lib
/usr/lpp/smq/samp

/usr/lib/nls/msg/prime/smq.cat
```

HP-UX

```
/opt/smq/bin
/opt/smq/include
/opt/smq/lib
/opt/smq/samp

/usr/lib/nls/msg/C/smq.cat
```

Installation

Sun

```
/opt/smq/bin
/opt/smq/include
/opt/smq/lib
/opt/smq/samp

/usr/lib/locale/C/LC_MESSAGES/smq.cat
```

Windows NT

```
\smq\bin
\smq\include
\smq\lib
\smq\samp

\smq\bin\smqcatnt.dll
```

Data conversion exit library

The data conversion exit library MQFMTSAP is shipped with SAP link. On the UNIX platforms a link is created for this library in `/usr/lib`. To avoid any problems when converting messages, make sure this is a directory in your LIBPATH when you run the inbound server.

Configuring MQSeries link for R/3

Before you can use MQSeries link for R/3 (SAP link), you must configure it to ensure that messages end up at the correct destinations. Before you can do any of this, you must understand the reasons for the configuration. These are explained in:

- "Understanding which MQSeries objects you need"
- "Advanced queue manager configuration" on page 19
- "Understanding RFC destinations" on page 20

The tasks themselves are described:

- "Task 1. Defining TCP/IP Ports for use with the operating system" on page 21
- "Task 2. Creating the MQSeries environment for SAP link" on page 21
- "Task 3. Defining the RFC destinations on R/3" on page 22
- "Task 4. Starting the servers" on page 29

When you have configured your system, you can then test the configuration, as described in "Testing the basic configuration" on page 31.

Understanding which MQSeries objects you need

You must define the MQSeries objects that SAP link is going to use before you can start either the inbound or the outbound server. You do this using the MQSeries commands (MQSC commands) supplied with MQSeries.

For more information about MQSeries:

- The *MQSeries System Administration Guide* for your platform provides information about setting up queue managers and running MQSC commands to create MQSeries objects.
- The *MQSeries Command Reference* provides information about the individual MQSC commands.
- The *MQSeries Distributed Queuing Guide* provides information about setting up channels between queue managers.

Defining a queue manager

You need to define some MQSeries queues that the SAP link servers use at run time. However, first you must define a **queue manager**, if you do not already have one. For an inbound server, you need to specify the name of the queue manager in the Message Queuing Options panel. The server automatically connects to the queue manager you specify and opens the queues it needs for input or output, as required.

Recommendation

Use the same queue manager for the queues used by both inbound and outbound servers. This is not a requirement, but it does simplify the configuration.

Configuring the link

MQSeries objects for an outbound server

For an outbound server, you must define two (MQSeries) queues:

outbound message queue
The queue that receives messages from R/3 applications. You must specify the name of this queue when you start the server using **smqso** command.

You can define this queue as a local queue or a remote queue (see "Choosing an outbound queue type" for more information).

outbound transaction ID queue
This is a local queue that SAP link uses to maintain a record of the transaction IDs within a unit of work. This queue is used by SAP link if it becomes necessary to back out a unit of work.

Choosing an outbound queue type

You can define your outbound queue as a local or remote queue.

If you are sending MQSeries messages between R/3 systems, you can use a remote queue definition that identifies the inbound message queue on the workstation that is going to receive the messages. This means that you do not need an application program to handle these messages. However, you do not have much flexibility with respect to the destination, because all messages go to the same place.

If you define the outbound queue as a remote queue, you must specify the following objects on the queue manager that is sending the MQSeries messages:

- The transmission queue for this remote queue. This is a local queue with its USAGE attribute specified as XMIT.
- A sender channel definition. Typically, your MQSeries messages are sent to another queue manager. To do this, you must define a sender channel on the local queue manager. This definition must be consistent both with the channel definition at the receiving end and with the definition of the transmission queue for the channel.

It is also possible to use remote queues in conjunction with queue manager aliasing (see the *MQSeries Distributed Queuing Guide* for more information on aliasing).

Figure 7 on page 18 shows an MQSeries configuration that uses a remote queue as the outbound queue.

Defining MQSeries objects for an inbound server

For an inbound server, you must define the following MQSeries queues on a suitable queue manager:

inbound message queue
The queue that receives messages from other applications.

inbound transaction ID queue
Typically, this is a local queue that SAP link uses to maintain a record of the transaction IDs within a unit of work. This queue is used by SAP link if it is necessary to back out a unit of work.

bad message queue
This queue is used to store inbound messages that the inbound server has attempted to process, but failed. For more information, see “Handling unrecognizable messages” on page 66.

Note: In the ini file, you can specify whether the server stops if a bad message is received.

You also need to define a receiver channel or equivalent. Typically, your inbound MQSeries messages come from another queue manager. To be able to receive these messages, you must define a receiver channel on the local queue manager. This definition must match the channel definition at the sending end. For more information about channels, see the *MQSeries Distributed Queuing Guide*.

Using the supplied definitions

SAP link provides a sample MQSC command file, `smqscdef.tst`. When you run this file, using the **runmqsc** command, it creates a set of queues corresponding to entries in the sample ini files supplied.

Note: This file does not support the example shown in Figure 7 on page 18.

Example of an MQSeries configuration

Figure 7 on page 18 shows one R/3 application sending information (in the form of IDocs) to an R/3 application on another system. In this example, the R/3 systems communicate using SAP link and MQSeries messages. This example shows how both the inbound and outbound servers can work in relation to MQSeries.

Configuring the link

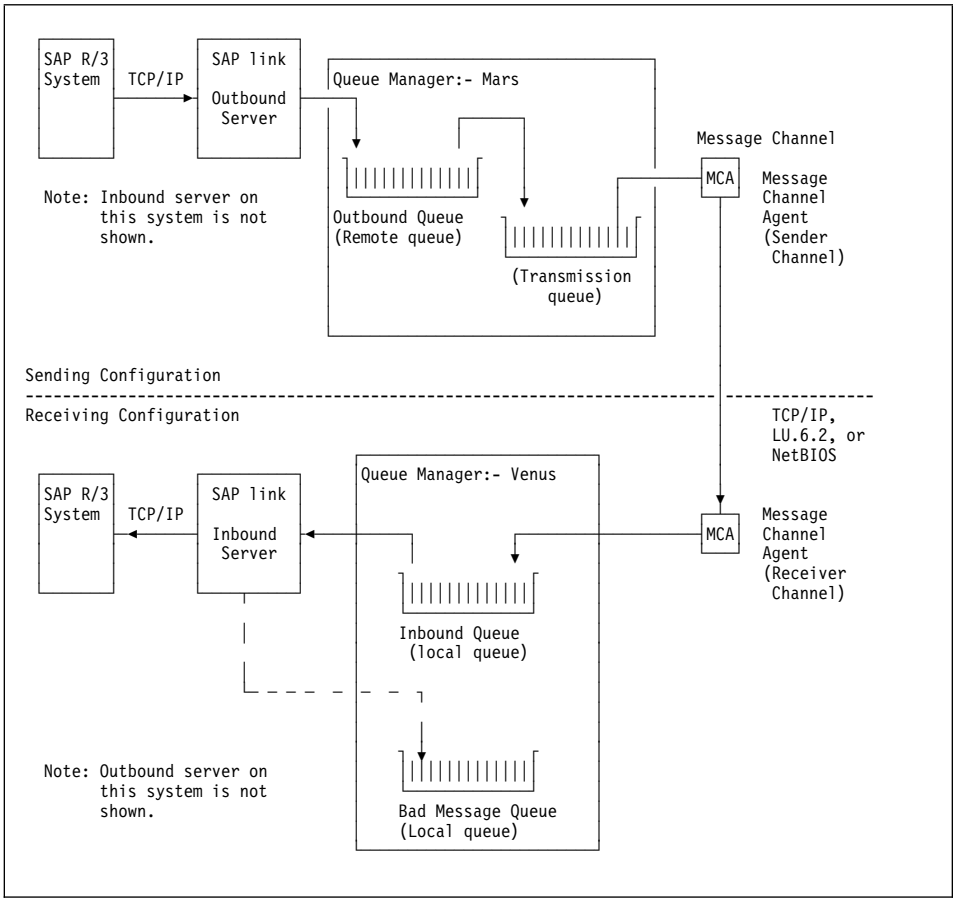


Figure 7. SAP link and MQSeries. You can send IDocs from one R/3 system to another using MQSeries messages. In this case, the SAP link outbound queue is defined as a remote queue that identifies a local queue on another queue manager. You must define the MQSC objects using the appropriate MQSC commands.

On the outbound server

In Figure 7 on page 18, the sending R/3 application sends IDocs to the SAP link outbound server that you have defined for it. The server creates an MQSeries message containing the IDocs as the application data part of the message. It also adds a link header to the message; this header contains information that the inbound server uses at the other end. The outbound server puts the MQSeries message it has created onto its outbound queue.

The outbound queue has been defined as a remote queue object (more accurately, a local definition of a remote queue). A remote queue object identifies a local queue on another system, that is, the inbound queue for the inbound server. You don't have to do this, but it is convenient because the message ends up on the other system, where it can be accessed by the inbound server on that system. Remember that MQSeries applications can put messages on local or remote queues, but can only retrieve messages (using MQGET) from a local queue.

On the inbound server

The inbound server decomposes the messages into its constituent IDocs and passes these on to the destination R/3 application, as specified in the SAP link header in the MQSeries message itself, or in the configuration information defined for the inbound server. Any messages the server cannot process are sent to the *bad message queue*. By making the appropriate statement in the inbound server ini file, you can specify whether the server stops when it gets a bad message.

Advanced queue manager configuration

In MQSeries, you can define more than one queue manager for each workstation on most platforms. SAP link allows you to specify different queue managers for each instance of a server; for outbound servers, you can specify different queue managers for different queues, as explained below.

Inbound servers

For an inbound server, all the queues for SAP link are owned by the same queue manager. These queues are:

- Inbound queue
- Inbound transaction ID queue
- Bad message queue

Note: You cannot use the same queue for different functions. You must define them as three separate queues, or SAP link cannot function properly.

Configuring the link

Outbound servers

For an outbound server, there are two queues for each instance of a server and you can, if necessary, put these on different queue managers. The queues for an outbound server are:

- Outbound queue
- Outbound transaction ID queue

You specify the name of the queue manager that owns the outbound queue in the *Queue manager name* field in the Message Queuing Options panel. You specify the name of the queue manager that owns the transaction ID queue, either as a parameter on the **smqso** (start outbound server) command or as a parameter in the outbound ini file. If you do not specify a queue manager explicitly, SAP link uses the default queue manager.

Note: You cannot use the same queue for both functions. You must define them as distinct queues, or SAP link cannot function properly.

Understanding RFC destinations

To enable an R/3 system to send information over SAP link, you must specify the RFC destination using an R/3 transaction in the ALE layer.

You use SAP link to transfer data, initially in the form of IDocs, from an R/3 system to another system, which can be:

- Another R/3 system
- An R/2 system
- A non-SAP system

On the outbound server, the IDoc data is put into an MQSeries message that is then put on the outbound server queue. You must identify this queue in the Message Queuing Options panel in the R/3 transaction, `sm59`.

If the final destination is another R/3 system, the IDocs contained in the message must be sent to the correct R/3 destination. To ensure this, you must specify the destination information either in the appropriate R/3 panel or in the ini file of the inbound server that is connected to the destination R/3 system.

If the destination is a non-SAP system, you do not have to specify the SAP destination information.

Specifying a remote R/3 client

The ALE layer on the sending R/3 system uses the parameters specified in the RFC Destination panels to connect to the outbound server. It uses the same principle whether the outbound server is local to the sending system or remote (across a network). Typically, there are multiple gateway servers across the network, all serving the sending R/3 system.

On one of these servers the *hostname* must be associated with a *program Id*, where *hostname* is the physical address of the machine on which the outbound server runs, and *program Id* is a unique parameter that links the sending R/3 system to a particular instance of the outbound server. Therefore, you have to specify this *program Id* in both the RFC destination panels and on the start server command.

Task 1. Defining TCP/IP Ports for use with the operating system

To allow the outbound server to communicate with the R/3 system, the following TCP/IP ports must be defined in the local system:

sapdpnn 32nn/tcp

sapgwnn 33nn/tcp

Where *nn* is the system instance number of your source R/3 system.

These TCP/IP port definitions must be placed into the following files according to your platform:

Platform	path/filename
AIX	/etc/services
HP-UX	/etc/services
Sun Solaris	/etc/services
Windows NT	...\WINNT35\system32\drivers\etc\services

Task 2. Creating the MQSeries environment for SAP link

Note: This procedure assumes that you are using ***one*** queue manager.

To create the MQSeries environment for SAP link:

1. If you have not already done so, create a queue manager for the queues (and channels) that SAP link requires.

   Use the **crtmqm** command to do this.

2. If the queue manager is not already running, start it with the **strmqm** command.
3. Create the queues and channels from an MQSC command file by running the following command against the default queue manager:

   ```
   runmqsc samp1.tst
   ```

   where `samp1.tst` is the file that contains the required MQSC DEFINE commands.

4. Start the channels for sending messages to, and receiving messages from, other queue managers.

Task 3. Defining the RFC destinations on R/3

This is where you start:

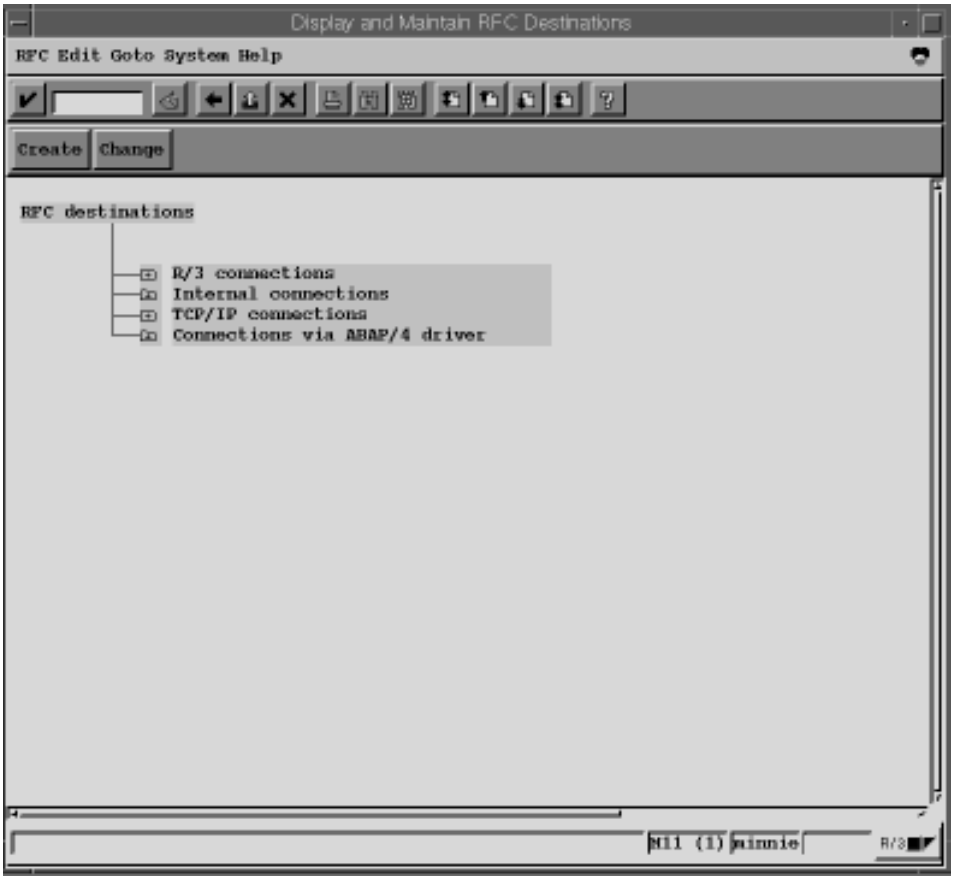


Figure 8. The SAP panel accessed through the sm59 R/3 transaction

To set up an outbound destination object from the ALE layer, use the SAP RFC configuration transaction (sm59).

1. In any R/3 panel, specify administration function sm59, by typing `/nsm59` in the command field. This changes the panel to that shown in Figure 8.
2. Click on the Create button in this panel to display the panel shown in Figure 9 on page 23.
3. Complete the fields in the RFC destination panel (shown in Figure 9 on page 23) as follows:
   a. In the RFC destination field, specify a destination name. This can be any name you like, but it must ***not*** contain any embedded spaces.
   b. Specify the Connection type as T (for TCP/IP). This is required.

c. Type some descriptive text for this connection in the *Description* field.

d. Save the settings. This displays the panel shown in Figure 10 on page 24.

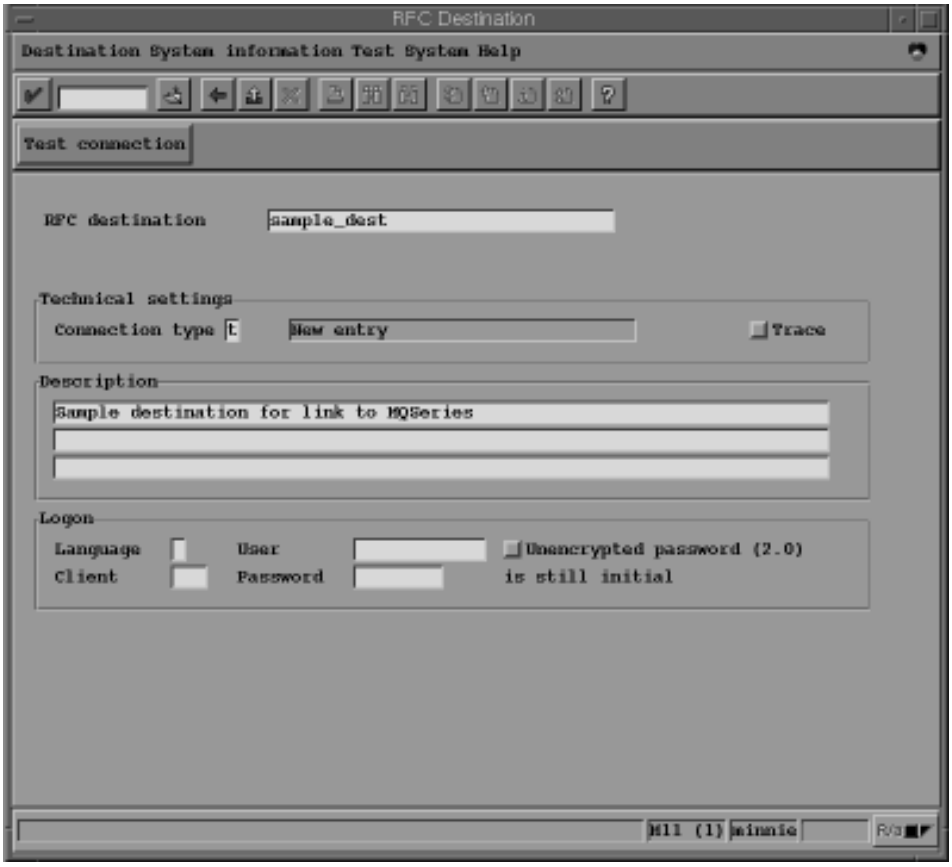


Figure 9. RFC Destination panel: specifying a destination and a connection type

4. Immediately click the Register button. This changes the appearance of the panel to that shown in Figure 10 on page 24.

Configuring the link

5. In the program ID field, type in a valid program ID. This is the name by which this particular outbound server connection will be known. You can call it what you like, but it must match the value you specify when you start the outbound server. The server gets the value from the command line or the outbound ini file. (The command line takes precedence over the ini file).

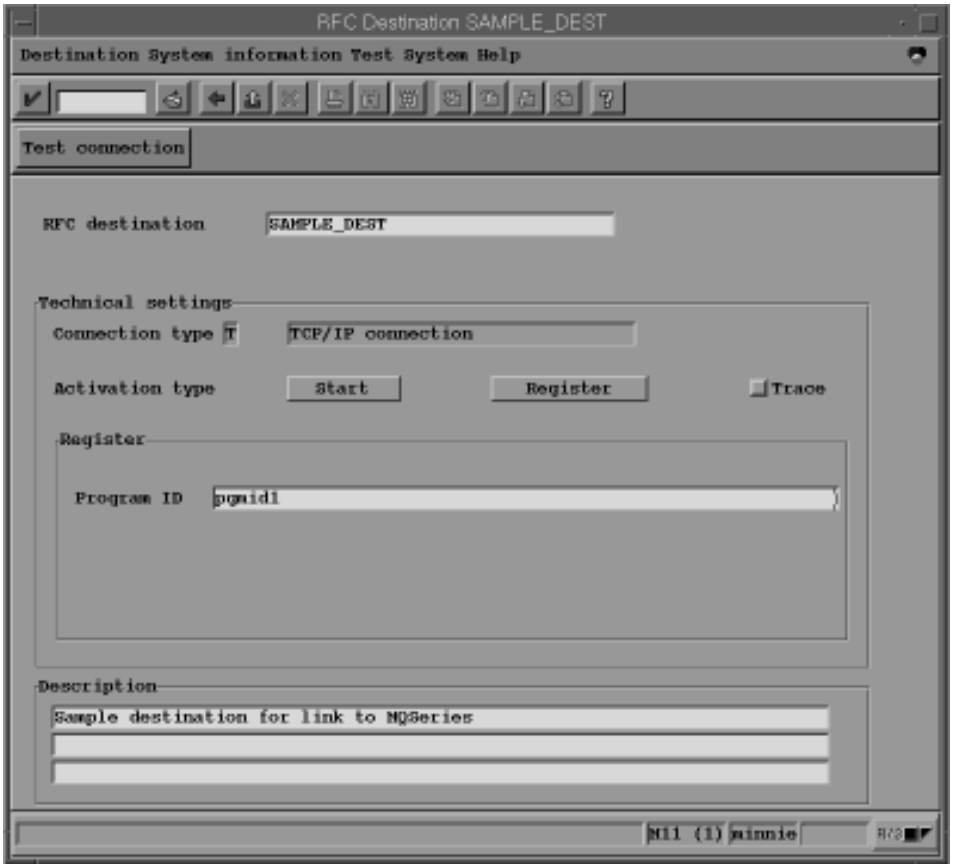


Figure 10. RFC Destination panel: specifying a program ID

6. Choose the Gateway options from the Destination menu (on the menu bar) to display the pop-up panel shown in Figure 11 on page 25.
7. In Figure 11 on page 25, specify values for the *Gateway host* and the *Gateway service* fields. These values should already be defined for your system. If you do not know what to put in here, talk to your SAP administrator.

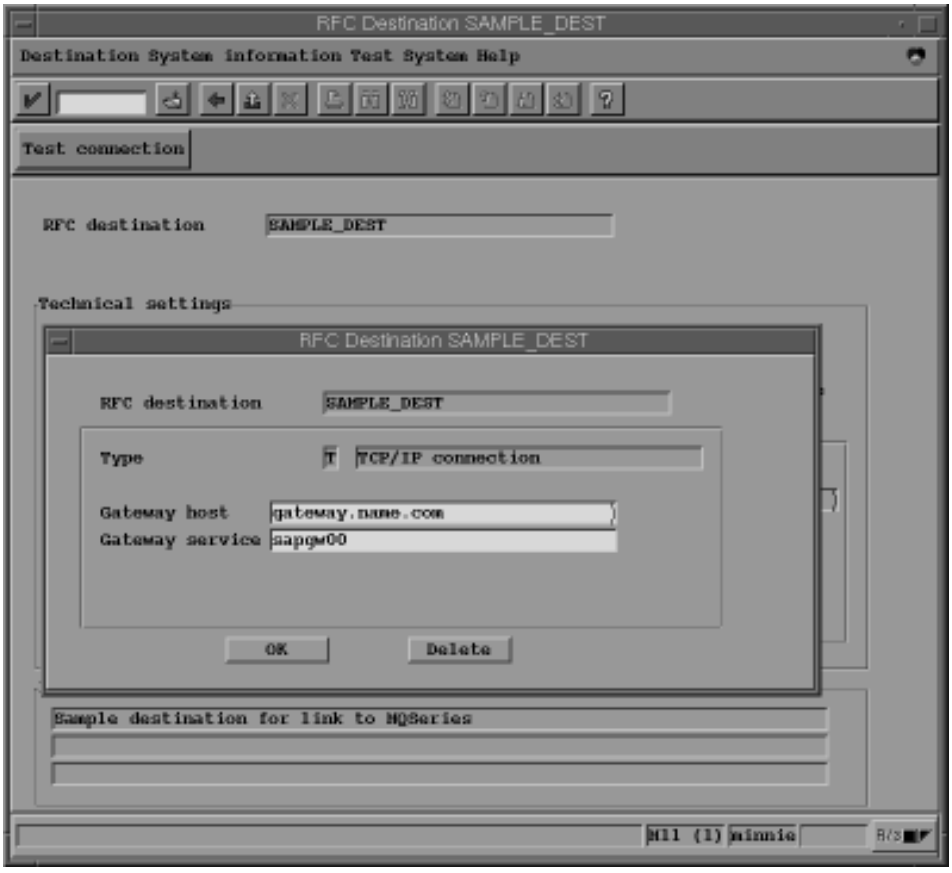


Figure 11. RFC Destination panel: specifying a gateway host and service

8. Occasionally the R/3 system may fail to connect to the SAP link outbound server, causing IDocs to be delayed. These IDocs are shown in R/3 transaction `sm58` with the message 'Gateway or Target system not active'. To force R/3 to automatically retry these IDocs, pull down the Destination menu and select TRFC options. The panel displayed in Figure 12 on page 26 will be shown. Fill in values for the number of retries and the retry interval. The values shown are examples. Some experimentation may be required to find values that provide optimum performance on your system.

   It is also recommended that you reduce the single connection attempts to the R/3 system by increasing the number of IDocs that can be placed in one MQSeries message. For example, package 10 IDocs together.

Configuring the link

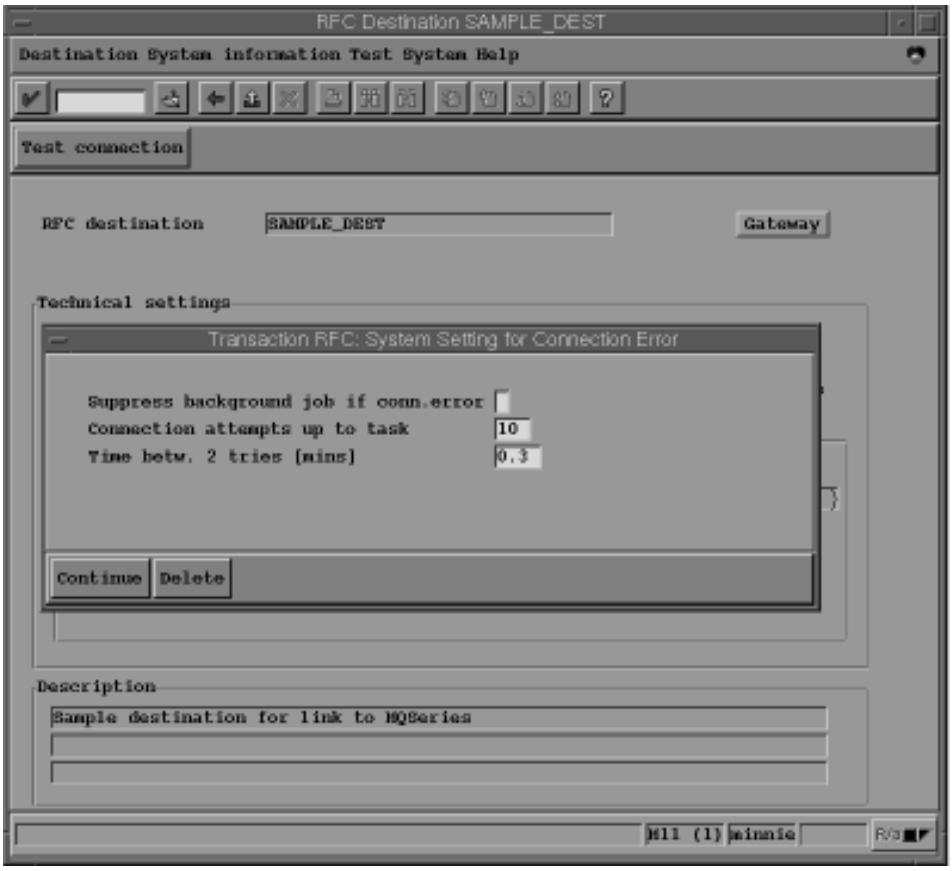


Figure 12. TRFC Options panel: specifying IDoc retry parameters

9. Choose the `MQS Options` option from the Destination menu to display the pop-up panel shown in “Using the Message Queuing Options panel” on page 27. Complete the fields in Figure 13 on page 27, as required. See “Using the Message Queuing Options panel” on page 27 for details.

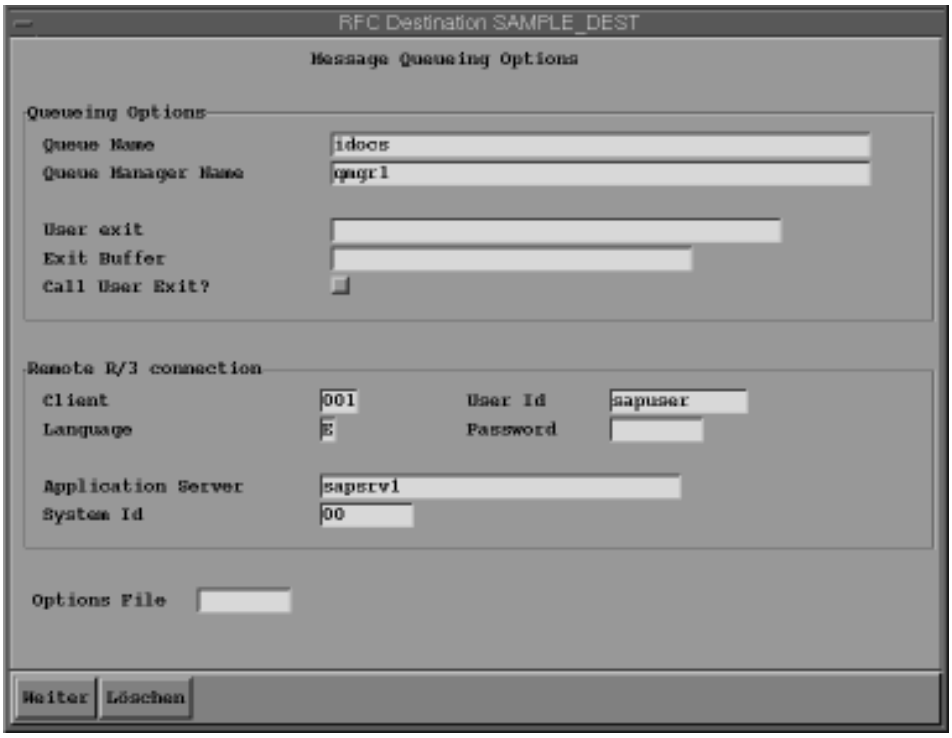


Figure 13. The Message Queuing Options panel for SAP link. Use this panel to specify Queuing options and the remote R/3 connection (if required).

Using the Message Queuing Options panel

Use the panel shown in Figure 13 to specify MQSeries information for an outbound server and R/3 destination information for the remote R/3 system.

Specifying the Queuing Options

In the **Queuing Options** fields, specify the following information for an outbound server configuration:

Queue Name

The name of the outbound message queue used by the outbound server. After generating the MQSeries message that includes the IDoc information, the outbound server puts the complete message on the outbound server queue.

This name is required.

Configuring the link

Queue Manager Name
The name of the queue manager that is associated with the outbound server and that owns the outbound server queue. The queue manager name can exist as an alias to the queue manager that owns the outbound queue. (See the *MQSeries Distributed Queuing Guide* for more information on queue manager aliasing.)

If you do not specify a name here, the queue manager name is taken from the -m option on the **smqso** (Start outbound server) command. If you omit the -m option, the name is taken from the ini file. If the queue manager name is not specified in any of these places, the default queue manager name is used.

User exit
The name and path of the user exit executable. To call the exit, you must also check the Call User Exit check-box.

Alternatively, if you type an asterisk (*) here and check the Call User Exit check-box, the user exit name in the ini file is used.

Exit Buffer
A flag containing up to 32 bytes of data that is passed to the user exit at run time. How you use this flag is up to you. For example, you can use this data to provide input data for a switch in the user exit where *test* tells the exit that this is a test run.

Call User Exit?
You must check this box if you want to call the user exit on the outbound server.

Specifying the remote R/3 connection parameters

If you are sending outbound messages to a remote R/3 system, you need to type in the parameters in the R/3 Information fields. Specify the following fields, as required:

Client
The client id of the destination R/3 system.

Language
The language identifier of the destination R/3 system.

User Id
The R/3 user ID of the destination R/3 system.

Password
The password of the destination R/3 system.

Application Server
The application server of the destination R/3 system.

System Id
The system instance number of the destination R/3 system.

Options File
Reserved for future use.

Notes:

1. If you do not know any of the R/3 connection parameters, contact your R/3 administrator.
2. Any fields you leave blank default to the values set on the inbound server on the remote system.
3. Much of this information is contained in the link header of the message. You should be aware that, if you define a password in the RFC destination panel, the password of the remote R/3 system is stored in this header. See “Security precautions” on page 71.
4. To create additional outbound destinations repeat the above procedure.
5. To test that the connection is working correctly, start the outbound server and, in the RFC destination panel, use the Test Communication button.
6. The program ID, gateway host, and gateway service uniquely identify an outbound server.
7. You can have multiple outbound servers.
8. While the outbound server is running you can see the registration of the outbound server destination object using transaction SMGW. This shows the TCP/IP address of the outbound server machine and the connection type.
9. If you make changes to the Message Queuing Options panel after IDocs have been created, and these IDocs are being collected as part of a pack, then the changes to the panel will affect all IDocs in the pack. That is, the changes are not just to the IDocs that were created after the changes were made. This is because the details from the options panel are only implemented when a complete pack of IDocs has been given to the outbound server.

Task 4. Starting the servers

You start both the inbound and outbound servers using the supplied commands. The servers can receive initialization information from three separate sources. When the same information is specified in more than one place, the priority is assigned as follows:

1. Command line parameters specified on the start server commands.
2. The initialization files. (You specify the appropriate ini file name on the start server commands.)
3. The RFC destination panels in R/3.

The commands to start the server are:

smqsi Start the inbound server.
smqso Start the outbound server.

Configuring the link

When you start an SAP link server, you can specify an ini file; the other parameters are optional. Therefore, the minimum forms of the commands are:

```
smqsi -iin.ini
smqso -iout.ini
```

where `in.ini` and `out.ini` are the respective ini files for the inbound and outbound servers.

To find out more about these commands and the parameters you can specify on them, see “Command reference” on page 47.

Using the initialization (ini) files

You can specify server configuration parameters in the ini file associated with that server. Typically, you use the ini files to set parameters that you do not change very often. You can specify a subset of these parameters on the command line.

Figure 14 shows the parameters in the sample ini file for an inbound server. To find out more about these parameters and what they do, see “Inbound server initialization parameters” on page 53 and “Outbound server initialization parameters” on page 58.

Note: The ini file is required for an inbound server, but optional for an outbound server.

```
client=100
user=hursley5
language=e
password=jennifer
sysnbr=00
hostname=9.165.255.88
transactionqueue=SMQ_INBOUND_TRANIDS
queuemanager=ehningen.qmgr
terminateonbadmessage=n
badmessagequeue=SMQ_BADMESSAGE_QUEUE
inboundqueue=SMQ_INBOUND_QUEUE
invokeexit=n
exitname=d:\saplink\bin\smqesmp1.dll
exitbuffer=AbelSeaman
maxconnections=256
rfctraceon=n
maxidocsize=0500000
```

Figure 14. Sample inbound ini file showing sample parameters

Initializing the outbound server

When you start the outbound server, the following parameters, specified in the R/3 destination panels, ***must match*** the equivalent parameters, specified in the ini file or in the command line parameters:

```
ProgramID
GatewayHost
GatewayService
```

Otherwise, the server will not be able to establish a successful connection to the R/3 system.

Initializing the inbound server

The inbound server is initialized in a similar way to the outbound server. The inbound server must connect to an R/3 system and to the queue manager that owns the queues associated with the server. To do this, it uses information that may be obtained from:

- The header of the message to be transmitted. This information, which is optional, is defined when you create an RFC destination.
- The command line parameters specified with the **smqsi** command.
- The inbound ini file.

Note: The outbound server puts the information in the SAP link header in the MQSeries message. This information is that specified in the Remote R/3 connection fields of the Message Queuing Options panel, described in “Specifying the Queuing Options” on page 27.

Testing the basic configuration

Note: *This section assumes an R/3 to R/3 communication over SAP link.*

You can now test the configuration by running through the following steps:

1. Start an outbound server, connecting it to the sending R/3 system.
2. Start an inbound server, connecting it to the destination (remote) R/3 system.
3. Send some IDocs.
4. Monitor the IDoc traffic.

1. Starting the outbound server

Check the three critical server initialization parameters:

1. ProgramID
2. GatewayHost
3. GatewayService

These must match the defined parameters in the R/3 RFC destination panel configuration for an outbound server.

Make changes, if necessary, to the outbound initialization file and then start the outbound server on the host machine using the command:

```
smqso -id:\saplink\out.ini
```

2. Starting the inbound server

Check that all the parameters in the initialization file point to valid objects in the R/3 and MQSeries configurations. Make changes, if necessary, to the inbound initialization file and then start the inbound server on the host machine:

```
smqsi -id:\saplink\in.ini
```

3. Sending the IDocs

To do this follow a procedure similar to the one described below for sending MATMAS IDocs (R/3 transaction `bd10`):

1. Select:

   Logistics ⇒ *Central functions* ⇒ *Distribution*

2. Select:

   Master data ⇒ *Material* ⇒ *Send*

3. Type in the following data (or its equivalent on your system):

```
Field          German Data          USA Data
Material          mq_coco           mq_coco
Message type       MATMAS           MATMAS
Logical system   MQTESTLSYS         MQTESTLSYS
```

4. Press Enter.
5. Return to the main menu.

4. Monitoring IDoc traffic

You can monitor the IDocs coming in and going out of an R/3 system using R/3 transaction `we05` or its equivalent:

1. Select:

   Logistics ⇒ *Central functions* ⇒ *Distribution*

2. Select:

   Monitoring ⇒ *IDoc overview*

3. Select the period of interest for IDoc traffic monitoring (the default is for today).
4. Press Enter.

Note: To view the status of any of the IDocs displayed in the colored table, double click on the IDocs of interest.

Writing user exits

This chapter tells you how to write user exits for the SAP link servers. It also describes the structures of MQSeries messages that contain IDocs and of the SAP link header. This chapter contains these sections:

For more information about MQSeries messages and their structure, refer to the *MQSeries Application Programming Reference*.

About user exits

SAP link provides a user exit facility that you can use to provide entry points to your own software routines. When enabled, the relevant exit handler calls the user exit for each MQSeries message that passes through the server.

What any user exit does depends on your requirements. All you need to do is write a suitable program for the function that you need, adhering to the rules described here. In particular, you must use the supplied return codes to tell the exit handler what to do next. For example, if a certain operation fails, and you want the user exit to end, you must issue a return code of SMQRC_TERMINATE_EXIT.

SAP link provides the source for two sample user exits, written in C, that you can modify to provide entry points to your own software routines. See "Sample user exits" on page 41 for more information.

When the exits are called

Calling the exits differs on the inbound and outbound servers.

User exits on the inbound server

If the exit is enabled, when the inbound server is started it calls the Initialize entry point, which sets up the exit. After this, whenever a message is got from the inbound message queue, the server calls the Execute entry point. After the data has been transferred to R/3, the server calls the Return entry point. Finally, when the server ends, it calls the terminate entry point.

Note: The execute and return functions are called from within an MQSeries unit of work (UOW). The retrieval of the message from the inbound queue is not committed until after the return function is completed. Initialize and terminate are called outside of an existing UOW.

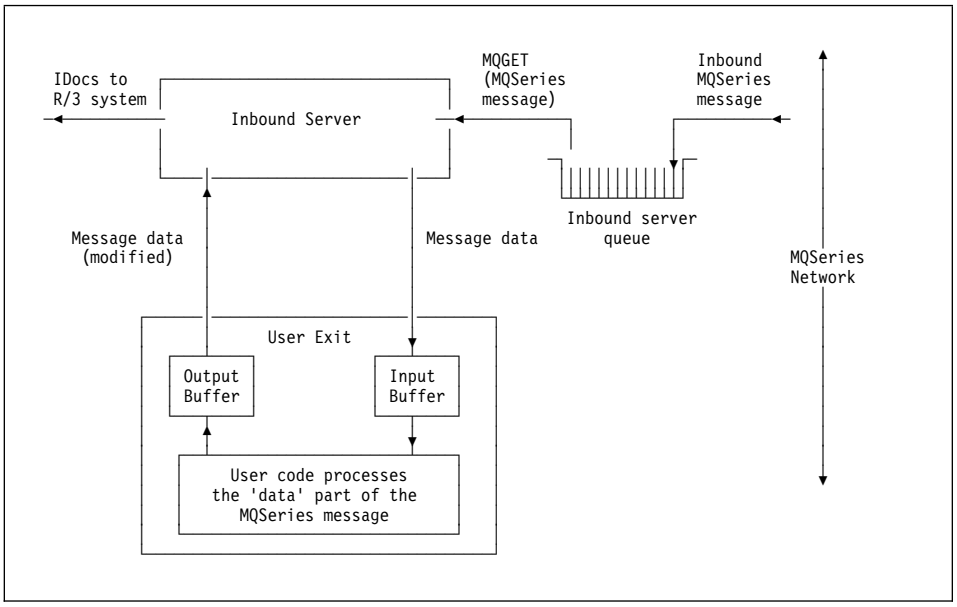


Figure 15. Data flow through the inbound server user exit after initialization

User exits on the outbound server

If the exit is enabled, when the outbound server is started it calls the Initialize entry point, which sets up the exit. After this, R/3 passes control to the exit. The first time the exit is called, the Initialize function is invoked to set up the exit. After this, the Execute and Return functions are called as before.

There can be multiple exits active on the outbound server. If an exit is specified in the ini file the initialize function is called when the server is started. If an exit is specified for a particular destination and the *Call User Exit* flag is set on the Message Queuing Options panel, the initialize function is called when the first data is received from R/3. After initialization, the execute function is called whenever data is received for a destination with an exit specified or implied before the MQSeries put to the output is attempted. The return function is called after the put.

The execute function is called without an existing MQSeries UOW. However, when the return function is called, an MQSeries UOW is in progress.

Figure 16 shows the outbound user exit being called ***before*** an MQSeries message is put onto the outbound message queue.

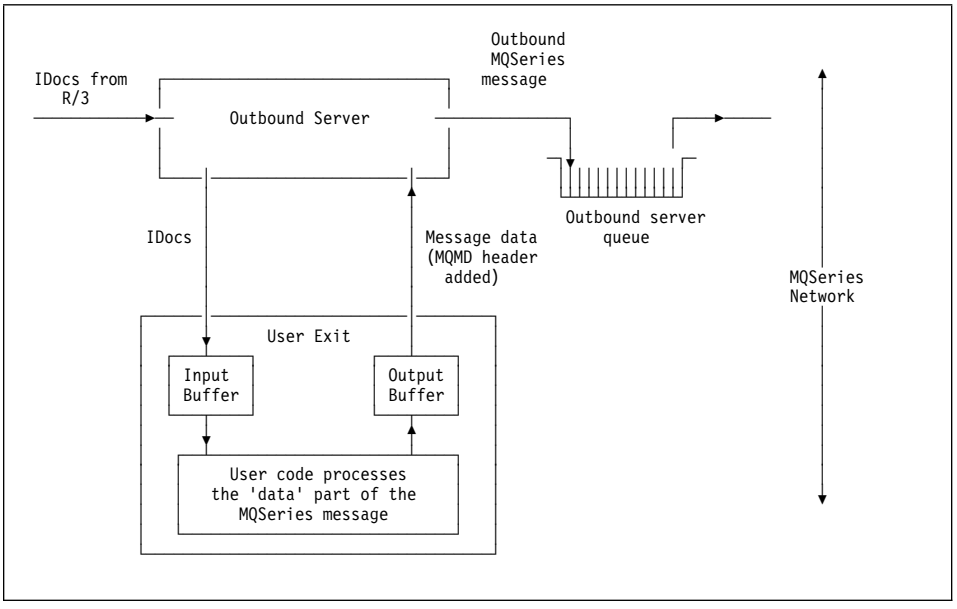


Figure 16. Data flow through the outbound server user exit

User exits

Enabling the user exits

The method you use to enable a user exit depends on the type of exit.

Enabling user exits for the inbound server

For the inbound server, you enable the exit by specifying `invokeexit=y` in the initialization file that you specify on the **smqsi** command. You must also specify a path and name for the executable. For example, if your initialization file for the inbound server contains these lines:

```
invokeexit=y
exitname=/home/your_name/exits/dothis.a
```

the exit `dothis.a`, in directory `/home/your_name/exits`, is called after the exit handler retrieves each MQSeries message from the inbound server input queue.

There is an optional exit buffer that you can use to pass your own user-defined data to the exit. You can specify this data using the `exitbuffer` parameter in the inbound ini file.

Enabling user exits for the outbound server

For the outbound server, you enable the exit by specifying values in the Message Queuing Options panel in R/3. See “Specifying the Queuing Options” on page 27 for details.

Note: If you want the outbound server to use the *exitname* and *exitbuffer* parameters in the outbound ini file, specify an asterisk (*) in the *Exit name* field and check the *Call User Exit?* box in the Message Queuing Options panel.

Understanding the external interface

The exit handler uses a defined set of call interfaces to communicate with the user exit. There are four entry points:

Entry point...	***What it does...***
Initialize	Sets up the user exit. This must be the first, or only, entry point ‘exported’ when linking the exit.
Execute	Typically, processes data.
Return	Cleans up any resources required by the Execute function.
Terminate	Terminates the user exit when the processing is complete.

Note: The entry point names are arbitrary; you do not have to use the names above although you will see them used in the samples.

The interface parameters are structures rather than individual values. Definitions of the data structures and the call interface are in the supplied include file `smqc.h`.

User exits

You must write all the user exit software in C and run it in the same process as the exit handler. Any cross-process calls and data sharing required by the user exit must be managed by the user exit software.

Recommendation

For complete information about the user exit, refer to the supplied sample user exit, which is located in the `samples` *sub-directory.*

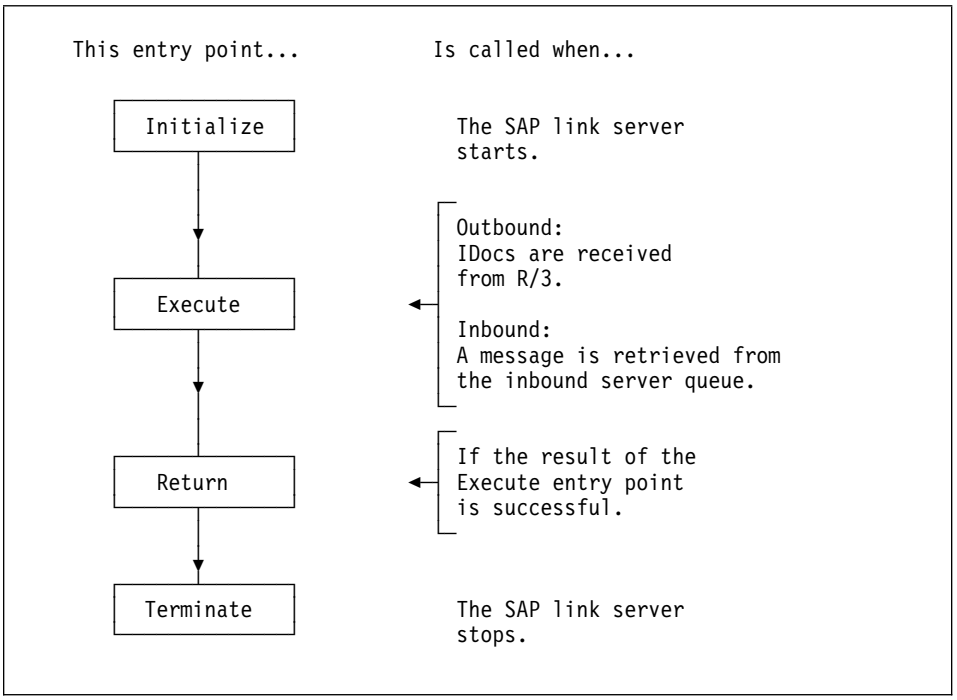


Figure 17. Entry points in a user exit. Initialize is the first entry point called.

User exits

The initialize entry point (smqUserExitInitialize)

The first entry point in your code must be of type `smqUserExitInitialize`.

This entry point enables the exit handler to pass setup information to the user exit. It also enables the user exit to perform its initialization procedures and to pass the addresses of the other entry points back to the exit handler.

The server calls this entry point with two parameters:

SMQEXIT_SETUP
This is a pointer to a structure that contains setup information for the user exit.

SMQEXIT_INIT
This is a pointer to a structure containing fields for the addresses of the remaining entry points and result code. Your user exit must supply the data for these fields:

Execute
Address of the execution entry point

Return
Address of the return entry point

Terminate
Address of the termination entry point

Execute entry point (smqUserExitExecute)

This is the call interface that executes the conversion procedures within your user exit. The server calls this entry point with three parameters:

SMQEXIT_SETUP
A pointer to a structure used with the call to the initialize procedure.

SMQEXIT_INPUT
Pointers to a structure containing information about the data to be converted and a pointer to the data (the input buffer). Your code must not alter the fields in this structure.

The pointers vary according to the direction of the message, inbound or outbound.

Header
On inbound this will be `NULL`.
On outbound this is a pointer to the `MQSAPH` structure as generated by the server.

MessageDesc
On inbound, a pointer to the MQMD associated with the retrieved message.
On outbound, a pointer to the MQMD that has been generated by the server.

MessageData
On inbound, a pointer to the message as retrieved from the inbound server queue.
On outbound, a pointer to the IDocs as received from R/3.

MessageDataLen
On inbound, the length of the message as retrieved from MQSeries.
On outbound, the length of the IDoc data as received from R/3.

Note: This does not include the length of the MQSAPH.

SMQEXIT_OUTPUT
A set of pointers including, for a successful conversion:

Header
On outbound, this should be NULL.
On inbound, this points to the MQSAPH structure generated by the exit.

MessageData
On outbound, this points to the message data generated by IDocs that are generated by the exit. This data is placed on the outbound queue.
On inbound, this points to the IDocs that are generated by the exit.

MessageDataLen
For both inbound and outbound, the length of the data addressed by MessageData.

Result
A return code indicating the result of the processing by your code. The return codes that you can use assume a data conversion operation (see “Return codes for the execution entry point”).

Return codes for the execution entry point

Depending on the results of your processing, you must specify one of the following return codes:

SMQ_CONVERT_OK
The conversion was completed without error.

The contents of the output buffer (that is, the results of processing by the exit) are inserted into an MQSeries message, which is then put on the appropriate server queue.

SMQ_CONVERT_NOT_NEEDED
Your code decided that the proposed conversion was not required.

The server will then act as though the exit was not called.

SMQ_CONVERT_FAIL
The proposed conversion was not completed successfully. The message will be put to the bad message queue if one has been specified.

Note: This may cause the server to end.

SMQ_TERMINATE_EXIT
The proposed conversion was not completed successfully, and you want to terminate the exit.

SMQ_TERMINATE_SERVER
The proposed conversion was not completed successfully, and you want to terminate this instance of the server.

Return entry point (smqUserExitReturn)

This entry point is called only if the result of the preceding execute procedure was successful.

When the exit handler has completed processing the data returned from the execution call, it calls the return entry point. The return enables the user exit to perform any cleanup, such as freeing memory.

The server calls this entry point with two parameters:

SMQEXIT_SETUP
A pointer to a structure that was used with the call to the initialize procedure.

SMQEXIT_OUTPUT
A pointer to a structure that was returned by the previous execution procedure. This structure also contains pointers to the memory allocated to contain the converted data in the user exit, the length of the converted data, in bytes, and a return code indicating the result of the data conversion.

Return codes for smqUserExitReturn

Depending on the results of your processing, you must specify one of the following return codes:

SMQRC_OK
SMQRC_TERMINATE_EXIT
SMQRC_TERMINATE_SERVER

The following return codes are not valid for this entry point:

SMQ_CONVERT_OK
SMQ_CONVERT_NOT_NEEDED
SMQRC_CONVERT_FAIL

Your code should not generate these return codes for this entry point.

Terminate entry point (smqUserExitTerminate)

The termination procedure is called when the server ends either normally or because of an error condition. The exit handler calls the termination procedure in the user exit to perform any cleanup, and terminate gracefully. The user exit must complete termination before returning from this call. The exit handler unloads the user exit when this call completes.

The server calls the termination entry point with two parameters:

SMQEXIT_SETUP
A pointer to a structure that was used with the call to the initialize procedure.

SMQEXIT_REQUEST
An enumerated type indicating the reason for the termination:

SMQ_REQUEST_EXIT
The exit returned SMQRC_TERMINATE_EXIT.

SMQ_REQUEST_SERVER
The exit should end because the server is terminating.

SMQ_REQUEST_ERROR
The exit should end because the server is terminating with an error.

Sample user exits

Two user exits are supplied that you can use as a basis for your own exits. One sample shows the exit structure, the other shows sending data to a non-R/3 system or receiving data from an R/3 system.

The source code for these exits is supplied in the `samples` directory.

Sample 1 (smqesmp1.c)

This sample shows the structure of a user exit. You can use it in either an inbound or an outbound server. For an inbound server, the exit expects the message data passed to it in this format:

```
<MQSAPH> <IDoc control> <IDoc data> and so on.
```

The sample allocates a buffer large enough to hold this data and passes this back to the exit handler. The data returned has separate pointers to the header and to the IDoc data. The Result field is set to `CONVERT_OK`. If the input data is not in the format expected, the exit returns with a result of `CONVERT_NOT_NEEDED`.

For an outbound server the exit gets a buffer large enough to hold both the header and the message data passed to it. It copies the header and the data to this buffer. This is then passed back to the exit handler in the output structure with a Result field set to `CONVERT_OK`.

Compiling user exits

Sample 2 (smqesmp2.c)

This sample shows the transformation of:

- An inbound message that is not in the R/3 IDoc format to the IDoc format required by the R/3 Materials Master function.
- An outbound message in IDoc format (as produced by the R/3 Materials Master function) to a user-defined format suitable for processing by a non-SAP system. This is the reverse of the inbound transformation. The user-defined formats are defined in the sample header file, `smqeshd1.h`.

This sample also shows the use of MQI calls within an exit.

Compiling user exits

The exits are dynamically loaded objects; they must have a name of no more than 8 non-blank characters. For example:

```
MYEXIT
```

Note: The MQSAPH header structure, which must be at the start of all SAP link messages, should be packed (aligned on 1-byte boundaries). The include file `smqc.h` defines a way to do this for each platform. To activate the correct alignment of this structure, you must define a compiler variable appropriate to your platform. The platforms and compiler variables are:

Platform	Variable
AIX	SMQ_AIX
HP-UX	SMQ_HPUX
SunOS	SMQ_SOLARIS
Windows NT	SMQ_NT

The following sample commands show how to specify these compiler variables.

For AIX:

```
cc -c I<Include> exit.c -DSMQ_AIX
ld exit.o -e Initialize -o MYEXIT -bM:SRE -H512 -T512 -L <Libraries>
```

For HP-UX:

```
cc -c -Aa +z -I<Include> exit.c -DSMQ_HPUX
ld -b exit.o -o MYEXIT +I Initialize -L <Libraries>
```

For SunOS:

```
cc -c -KPIC -I<Include> exit.c -DSMQ_SOLARIS
ld -G exit.o -o MYEXIT
```

The samples assume a name of `Initialize` for the initialize entry point.

For Windows NT:

```
cl /nologo /W3 /GX /Zi /YX /Od /D "WIN32" /EXIT.C /D "SMQ_NT"
link /nologo /SUBSYSTEM:console \
               /INCREMENTAL:yes /PDB:"MYEXIT.PDB" \
               /MACHINE:I386 /OUT:MYEXIT.DLL
```

Message formats

To send transactions to or from R/3, SAP link uses MQSeries messages, that is, messages that have the same format as those used in MQSeries. Figure 18 shows the structure of a message, broken down into its constituent levels.

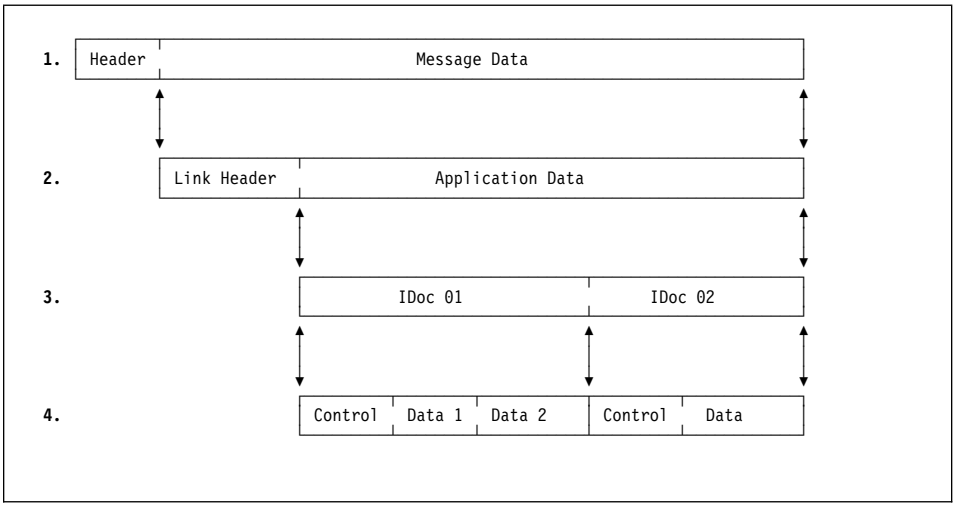


Figure 18. MQSeries message structure and IDocs

1. This is the generic MQSeries message format. The header contains MQSeries control information for processing the message, and is followed by message data.
2. The message data begins with a link header that is used by the SAP link software (and in some cases by non-SAP applications). The link header contains information to identify the destination of the message. Figure 19 on page 44 shows the details of the SAP link header structure.

   The link header is followed by application data. Typically, for inbound transactions, the application data field is in IDoc format.

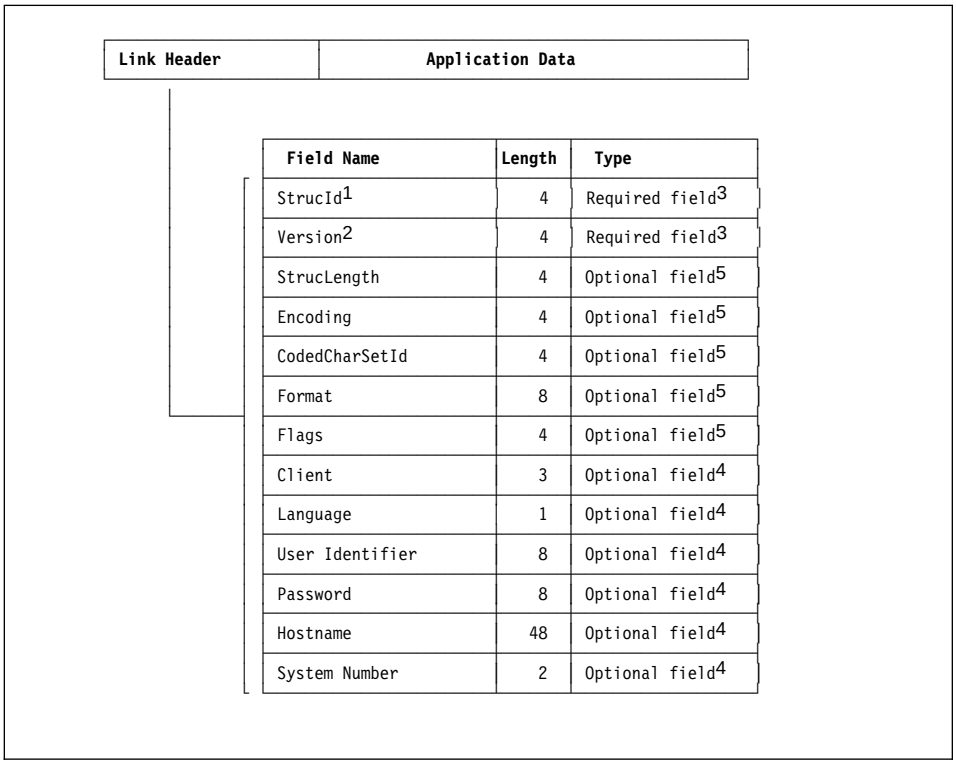
   For outbound transactions the application data field should be converted to the format required by the target application. However, if the target application is another R/3 system, this field takes an IDoc format.
3. In an MQSeries message, the application data consists of one or more IDocs stored contiguously with no separators between them. The maximum size of the application data is specified in the ini file, subject to a maximum of 4MB for MQSeries messages. See “Other defaults for the inbound server” on page 57 for more information.

4. Each IDoc has a control table and one or more data tables of fixed length.

   Each IDoc is identified by a unique IDoc number that is stored in a field in both the control and data tables. A new IDoc within the same message is identified by a new IDoc number contained in the control table. Figure 18 on page 43 shows a message with two IDocs. The first has two data parts and the second has one data part.

SAP link header structure

The SAP link header contains information about the message, including information about the remote R/3 destination of the message. The outbound server automatically inserts an SAP link header in every outbound message. If an application is sending a message inbound R/3 using SAP link, the sending application must construct this header.



Field Name	Length	Type
StrucId[1]	4	Required field[3]
Version[2]	4	Required field[3]
StrucLength	4	Optional field[5]
Encoding	4	Optional field[5]
CodedCharSetId	4	Optional field[5]
Format	8	Optional field[5]
Flags	4	Optional field[5]
Client	3	Optional field[4]
Language	1	Optional field[4]
User Identifier	8	Optional field[4]
Password	8	Optional field[4]
Hostname	48	Optional field[4]
System Number	2	Optional field[4]

Figure 19. Structure of the SAP link header

Notes:

1. StrucId, the structure identifier, is:

   ```
   SAPH
   ```

2. The version number is:

   ```
   1L
   ```

   That is, the value 1 in a long integer field.

3. StrucId and Version are required fields.

4. If you leave blank any optional fields in the R/3 Message Queuing Options panel, the outbound message is sent without containing enough information to identify the remote R/3 destination (although it can find the destination queue from the MQMD header). In this case, the remote R/3 destination is determined at the inbound SAP link server from the initialization (ini) file that was used to start this instance of the server.

5. The user has the option of using the other entries in the SAP link header structure. These may be of use when communicating between R/3 and non-SAP systems.

   StrucLength Length of the SAP link header structure.

   Encoding Native machine encoding.

   CodedCharSetId Native machine coded character set Id.

   Format Format name. The rules for encoding the format name are identical to the rules for encoding the format field in the MQMD structure.

   Flags Reserved.

Constructing messages with IDocs

If you send data to an inbound server from a source outside R/3, you must construct an MQSeries message that contains the data in IDoc format, with control and data sections, as shown in Figure 18 on page 43.

If you have more than one IDoc in each message, you must ensure that each IDoc has a 16-byte unique identifier starting at position 13 of the control section of each IDoc. Otherwise, the inbound server treats the whole package as one IDoc and reports error message `SMQ4172 - Invalid length for IDoc`.

If your IDocs originate from R/3, you should specify the identifiers in the R/3 Message Queuing Options panel.

Message formats

If you are sending messages to the inbound queue from a platform with a different code page, then code page conversion is required. The inbound server uses the MQSeries get option, MQGMO_CONVERT, and automatically calls the supplied data conversion exit 'MQFMTSAP' if the message has the format field in the MQMD structure set to 'MQFMT_SAP'. The outbound server sets this automatically, but if you use your own program to put messages to the inbound queue, you need to set the format field in the MQMD structure to MQFMT_SAP when you do the put.

See the chapter on user exit programs in the *MQSeries Distributed Queuing Guide* for more information about data conversion exits.

Note: When compiling your own programs to construct messages that are to be transferred from MQSeries to R/3, you must ensure that the MQSAPH header structure is packed (aligned on 1-byte boundaries). See the note on page 42 for more details.

Command reference

This chapter describes the commands that you can use to start either the inbound server or outbound server.

smqsi Start the inbound server; see page 48.
smqso Start the outbound server; see page 50.

Initialization parameters

When you start a server, it reads the command parameters and the contents of the initialization (ini) file specified by the -i parameter on the command line. If the same parameters are defined on both the command line and in the ini file, the command line parameters are the ones that are actually used.

As far as possible, use the command line to specify particular values of parameters; use the ini files for frequently-used values.

smqsi (Start inbound server)

Use the **smqsi** command to start the SAP link inbound server, allowing information to flow inbound into an R/3 system from a specified MQSeries queue.

When you issue this command, it must contain an **-i***InitializationFileName* parameter, or the server will not start.

Command line parameters take precedence over any parameters you specify in the ini file.

To stop the inbound server, type CTRL+C.

Syntax

```
smqsi  -iInitializationFileName
       -mQueueManagerName
       -qInboundQueueName
```

Required parameters

-i*initializationFileName*

Specifies the full path and file name of the initialization file. This file contains parameters that customize the execution of the inbound server.

See “Inbound server initialization parameters” on page 53 for more information.

Optional parameters

-m*QueueManagerName*

Specifies the name of the MQSeries queue manager that owns the MQSeries objects specified for the inbound server. If specified, this parameter overrides the `QueueManagerName` parameter, if any, specified in the inbound ini file. If you do not specify a queue manager in any of these places, the default queue manager on the host running the inbound server is used.

The specified queue manager must be running, or the server cannot start.

-q*InboundQueueName*

Specifies the queue name used to store messages being transferred to the R/3 system. This queue must have already been created before the start command is called.

You must specify a valid inbound queue name on either the command line or in the ini file, or the server cannot start.

Examples

1. Start the inbound server and take all initialization data from the `myini.ini` file.

   ```
   smqsi -imyini.ini
   ```

2. Start the inbound server, reading the initialization data from the specified file `in.ini`, except for the inbound queue, which is to be `inbound.queue`. If the ini file has an entry for the *InboundQueueName* parameter, this is ignored.

   ```
   smqsi -id:\saplink\in.ini -qinbound.queue
   ```

smqso (Start outbound server)

Use the **smqso** command to start the SAP link outbound server. The outbound server takes one or more IDocs from an R/3 system, converts them to an MQSeries message, and puts this message on the (MQSeries) outbound message queue named in the ini file or on the command line.

To start the outbound server, the *gateway service, gateway host,* and *program Id*, must be defined at least once in the ini file or on the command line. If you specify a parameter in both the ini file and on the command line, the command line parameters take precedence.

To stop the outbound server, type CTRL+C.

Syntax

```
smqso  -iInitializationFileName
       -xGatewayService
       -gGatewayHost
       -aProgramId
       -mQueueManagerName
```

Optional parameters

-i*InitializationFileName*

Specifies the full path and file name of the initialization (ini) file for the outbound server. This file contains parameters to customize the execution of the outbound server. See “Outbound server initialization parameters” on page 58 for more information.

-x*GatewayService*

Specifies the gateway service used by the R/3 connection. You must specify a valid gateway service either on the command line or in the ini file. The name of the gateway service used with the command must match the name specified in the R/3 RFC destination panel. Otherwise, the outbound server cannot complete its startup procedure.

-g*GatewayHost*

Specifies the gateway host used by the R/3 connection. You must specify a valid gateway host either on the command line or in the ini file. The name of the gateway service used with the command must match the name specified in the R/3 RFC destination panel. Otherwise, the outbound server cannot complete its startup procedure.

-a*ProgramId*

A unique identifier that specifies which host is to receive the outgoing R/3 information. The name of the program ID used with the command must match the name specified in the R/3 RFC destination panel. Otherwise, the outbound server cannot complete its startup procedure.

-m*QueueManagerName*

Specifies the name of the MQSeries queue manager that owns the MQSeries objects used by the outbound server. This parameter overrides the `queuemanagername` parameter, if any, specified in the outbound ini file.

If you do not specify a queue manager name on either the command line or in the ini file, the default queue manager on the host running the outbound server is used.

The specified queue manager must be running, or the server cannot start.

Examples

1. Start the outbound server, taking all its initialization data from the ini file `out.ini`:

   ```
   smqso  -id:\saplink\out.ini
   ```

2. Start the outbound server reading the initialization data from the ini file `out.ini`. Then override the *ProgramId*, *GatewayHost*, and *GatewayService* parameters in the file with those specified on the command line below. This allows the server to connect to a different preconfigured R/3 RFC destination containing, in this case, the parameters `prog_id_1`, `Host_1`, and `sapgw45`.

   ```
   smqso  -aprog_id_1 -gHost_1 -xsapgw45 -id:\saplink\out.ini
   ```

3. Start the server, reading all the initialization data from the specified file and then override the *QueueManagerName* parameter. This server uses `saturn.queue.manager` as the destination queue manager.

   ```
   smqso -id:\saplink\myini.ini -msaturn.queue.manager
   ```

smqso

Initialization files

The chapter tells you about the initialization (ini) files that are used when you start one or other of the SAP link servers. Using an ini file is a convenient way to specify parameters that you use repeatedly. Also, using the default definitions minimizes the time you need to spend on configuration.

There are two different types of ini file, one for each server type. When you start the inbound server, you must specify the name of the ini file using the -i option on the **smqsi** command. For the outbound server, specifying an ini file on the **smqso** command is optional. For more information about these commands, see “Command reference” on page 47.

Inbound server initialization parameters

You can create an inbound initialization (ini) file to specify parameters that are used when you start the inbound server (using the **smqsi** command).

Note: Any parameters you specify in the ini file are overwritten if you specify the same parameters on the command line.

For the inbound server, there are four categories of parameters:

1. Remote R/3 system connection parameters
2. MQSeries definitions used by the inbound server
3. User exit information
4. Other defaults

R/3 system connection defaults

The following parameters specify the defaults for the destination information of inbound messages. The inbound server tries to establish a connection on receiving an inbound message using these connection variables.

Parameter	**client**
Example	client=100
Description	Specifies the client number of the R/3 system for inbound server connection; that is, the destination of the messages.
Maximum length	3 characters
Format	Conforms to R/3 naming conventions; always a number.

Inbound ini files

Parameter	**user**
Example	user=fred
Description	Specifies the R/3 logon user name for an inbound server connection to the remote R/3 system
Maximum length	12 characters
Format	Conforms to R/3 naming conventions.

Parameter	**language**
Example	language=e
Description	Specifies the language of the R/3 system for inbound server connection to the remote R/3 system
Maximum length	1 byte
Format	Conforms to R/3 naming conventions: always a letter.

Parameter	**password**
Example	password=jennifer
Description	Specifies the R/3 user password for the inbound server connection to the remote R/3 system
Maximum length	8 characters
Format	Conforms to R/3 naming conventions.

Parameter	**sysnbr**
Example	sysnbr=00
Description	Specifies the instance number of the R/3 system for inbound server connection to the remote R/3 system.
Maximum length	2 characters
Format	Conforms to R/3 naming conventions.

Parameter	**hostname**
Examples	hostname=9.165.255.88 hostname=abletasman
Description	The application server name used by the R/3 system for an inbound server connection to the remote R/3 system
Maximum length	48 characters
Format	Conforms to R/3 naming conventions.

MQSeries definitions used by the inbound server

These parameters specify the name of the MQSeries queue manager and the queues to be used with the inbound server. In addition, one of the parameters specifies whether the server is to be stopped if it receives a message in a format unrecognizable to R/3. (For more information about unrecognizable messages, see “Handling unrecognizable messages” on page 66.)

Parameter	**transactionqueue**
Examples	transactionqueue=red.local.queue transactionqueue=SMQ_INBOUND_TRANIDS
Description	Specifies the name of the queue used by the inbound server to store transaction ids. The default, if you do not specify anything, is specified as SMQ_INBOUND_TRANIDS.
Maximum length	48 characters
Format	Conforms to the MQSeries queue name specifications.

Parameter	**queuemanager**
Example	queuemanager=saturn.queue.manager queuemanager=ehningen.qmgr
Description	Specifies the name of the queue manager that owns the inbound queue object (and the inbound transaction queue). If you do not specify a name here, the local default queue manager is used, if one exists.
Maximum length	48 characters
Format	Conforms to the MQSeries queue manager name specifications.

Parameter	**terminateonbadmessage**
Examples	terminateonbadmessage=y terminateonbadmessage=yes terminateonbadmessage=n terminateonbadmessage=no
Description	Specifies whether the inbound server is stopped if it receives a message that is unrecognizable. The default is that the server is not stopped.
Maximum length	3 characters
Format	y, n, yes, no

Inbound ini files

Parameter	**badmessagequeue**
Examples	badmessagequeue=SMQ_BADMESSAGE_QUEUE badmessagequeue=bad.message.queue
Description	Specifies the name of the bad message queue. The server puts any MQSeries messages that it does not recognize on this queue.
Maximum length	48 characters
Format	Conforms to the MQSeries queue name specifications.

Parameter	**inboundqueue**
Examples	inboundqueue=yellow.local.queue inboundqueue=SMQ_INBOUND_QUEUE
Description	Specifies the name of the inbound message queue. The inbound server retrieves incoming messages from this queue and converts them to IDocs for input to the receiving R/3 system.
Maximum length	48 characters
Format	Conforms to the MQSeries queue name specifications.

User exit information

These parameters specify whether the named user exit is called by the inbound server.

Parameter	**invokeexit**
Examples	invokeexit=y invokeexit=n
Description	A flag that specifies whether the named exit is to be used. The default is n.
Maximum length	1 character
Format	y or n

Parameter	**exitname**
Examples	exitname=d:\saplink\bin\smqesmp2.dll exitname=/usr/ableman/saplinkbin/smqesmp2.a
Description	The name of the user exit DLL or shared library to be used by the exit handler. If you do not specify a path here, you must specify one in the PATH environment variable for your system.
Maximum length	40 characters
Format	Conforms to operating system format for path and file name.

Parameter	**exitbuffer**
Example	exitbuffer=Luzern
Description	Specifies the optional contents of a buffer passed to the user exit on initialization. You can use this parameter to define a switch to give different user exit functions depending on these contents.
Maximum length	32 characters
Format	Free form text.

Other defaults for the inbound server

Parameter	**maxconnections**
Example	maxconnections=256
Description	Specifies the maximum number of cached handles (recorded connections) from the inbound server to R/3 systems. The default is 256.
Maximum length	4 characters
Format	A number.

The following parameters specify other defaults used by the outbound server:

Parameter	**rfctraceon**
Examples	rfctraceon=yes rfctraceon=y rfctraceon=no rfctraceon=n
Description	Option to turn the RFC trace on or off. The trace file is saved in `dev_rfc` in the current directory. By default, tracing is switched off.
Maximum length	3 characters
Format	y, n, yes, no

Parameter	**maxidocsize**
Examples	maxidocsize=0500000 maxidocsize=2000
Description	Specifies the maximum size, in bytes, of an inbound IDoc batch. The default maximum size is 2 097 152 bytes.
Maximum length	7 characters
Format	A number in the range 1 to 4 186 112 inclusive.

Outbound ini files

Outbound server initialization parameters

You can create an outbound initialization file to specify parameters that are used when you start the outbound server (using the **smqso** command).

Note: Any parameters you specify in this file are overwritten if you specify the same parameters on the **smqso** command.

For the outbound server, there are four categories of parameters:

1. RFC destination parameters
2. MQSeries definitions
3. User exit information
4. Other defaults

RFC destination parameters

The server can establish a connection with an RFC destination, but only if the RFC Destination parameters match the following three parameters.

Parameter	**gatewayhost**
Example	gatewayhost=ha0016
Description	Name of the R/3 gateway host to connect.
Maximum Length	48 characters
Format	Conforms to R/3 naming conventions.

Parameter	**gatewayservice**
Example	gatewayservice=sapgw86
Description	Name of the R/3 gateway service residing on the named gateway host.
Maximum length	12 characters
Format	Conforms to R/3 naming conventions.

Parameter	**programid**
Example	programid=telstaret
Description	Name of the unique program ID parameter.
Maximum length	24 characters
Format	Conforms to R/3 naming conventions.

Outbound user exit defaults

These defaults are used when you type an asterisk (*) in the **Exit Name** field in the R/3 Message Queuing Options panel.

Parameter	**exitname**
Examples	exitname=d:\saplink\bin\smqesmp1.dll exitname=/usr/ableman/saplinkbin/smqesmp1.a
Description	The name of the user exit DLL or shared library to be used by the exit handler. If you do not specify a path here, you must specify one in the PATH environment variable for your system. This exit name and path are used if you check the *Call User Exit* box in the Message Queuing Options panel and you also specify an asterisk (*) in the **Exitname** field.
Maximum length	40 characters
Format	Conforms to operating system format for path and file name.

Parameter	**exitbuffer**
Example	exitbuffer=LosAndos
Description	Specifies the optional contents of a buffer passed to the user exit on initialization. You can use this parameter to define a switch to give different user exit functions depending on these contents.
Maximum length	32 characters
Format	Free form text.

MQSeries definitions

The following parameters define the names of the queue manager and the transaction queue.

Note: The outbound message queue name is defined only in the RFC destination panels in R/3.

Parameter	**transactionqueue**
Example	transactionqueue=orange.local.queue
Description	Specifies the name of the queue used by the outbound server to store transaction ids. If you do not specify this parameter, the queue name SMQ_OUTBOUND_TRANIDS, as defined in outbound server code, is used as the default.
Maximum length	48 characters
Format	Conforms to the MQSeries queue name specifications.

Outbound ini files

Parameter	**queuemanager**
Examples	queuemanager=MN01 queuemanager=saturn.queue.manger
Description	Specifies the name of the local queue manager that owns the transaction id queue objects. (This may also be, but not necessarily, the same queue manager that owns outbound server queue.) If you do not specify a name here, the local default queue manager is used if there is one.
Maximum length	48 characters
Format	Conforms to the MQSeries queue manager name specifications.

Other defaults for the outbound server

The following parameters specify other defaults used by the outbound server.

Parameter	**rfctraceon**
Examples	rfctraceon=yes rfctraceon=y rfctraceon=no rfctraceon=n
Description	Option to turn trace on or off. The trace file is saved in `dev_rfc` in the current directory. By default, tracing is switched off.
Maximum length	3 characters
Format	y, n, yes, no

Parameter	**maxidocsize**
Example	maxidocsize=500000
Description	Specifies the maximum size of an outbound IDoc batch. The default is 2 097 152 bytes.
Maximum length	7 characters
Format	Any number between 1 and 4 186 112 inclusive.

Parameter	**usesapuserid**
Examples	usesapuserid=n usesapuserid=no usesapuserid=y usesapuserid=yes
Description	Specifies whether the R/3 user ID is used in the **UserIdentifier** field of the MQMD message header. See also “Security precautions” on page 71.
Maximum length	3 characters
Format	y, n, yes, no

Outbound ini files

Troubleshooting

This chapter contains information to help you to diagnose and solve problems encountered when using SAP link.

The contents of this section include:

Outbound server problem diagnosis

If an error occurs when the outbound server is processing a message, the following list will help you to diagnose the error:

1. Check that the local queue manager is started.
2. Are the outbound and transaction id queues defined and usable?
3. Do you have access to all queues used by the outbound server?
4. Check that the R/3 RFC destination *gateway host name, gateway service,* and *program ID* match the same parameters defined at outbound server initialization.
5. Are the TCP/IP ports defined?
6. Check that sufficient memory has been allocated to handle the size of the processed IDoc batches.
7. If an exit is used, check that the exit has executed successfully.
8. On R/3, check that the ALE configuration is correct.
9. On R/3, verify that the "Partner Profile" configuration (R/3 transaction code `we20`) has the batch IDoc processing option selected.
10. Ensure that the R/3 database and R/3 instance are active for the outbound R/3 system.
11. On R/3, check that IDocs have not been routed unnecessarily to the R/3 transaction `sm58`.

You may also find further diagnostic information through the MQSeries home page on the World Wide Web. Refer to "Information about MQSeries on the Internet" on page viii for further information.

Troubleshooting

Inbound server problem diagnosis

On the inbound server, you can get invalid IDoc length errors (SMQ4172) if you do not ensure that each IDoc with an MQSeries message has a unique identifier. See “Constructing messages with IDocs” on page 45 for details.

You may also receive unrecognizable, or bad, messages on the inbound queue. For more information about handling these messages, see “Handling unrecognizable messages” on page 66.

If an error occurs when the inbound server is processing a message, you can use the following list to help diagnose the error:

1. Check that the local queue manager is started.
2. Verify that the inbound, transaction id, and bad message queues are defined and usable.
3. Do you have access to all the queues used by the inbound server?
4. Has enough memory been allocated to handle the size of messages being processed?
5. If an exit is used, verify that the exit completed successfully.
6. Check that the following R/3 connection parameters provide a connection to the required remote R/3 system:
   - Client
   - User
   - Password
   - Language
   - System instance number
   - Application server host name
7. Check the bad message queue for messages (refer to “Handling unrecognizable messages” on page 66.)
8. On R/3, check transaction `we05` for processed inbound IDocs.

You may also find further diagnostic information through the MQSeries home page on the World Wide Web. Refer to “Information about MQSeries on the Internet” on page viii for further information.

Communication and system failures

Communication between SAP link and R/3 uses TCP/IP network protocol services and therefore any failure with the TCP/IP network should be referred to your network administrator. However, with the assistance of the checklists given earlier in this chapter, the user can ensure that the problem is not connected with the SAP link configuration before passing the problem to the TCP/IP administrator.

SAP link can detect problems with connection to the R/3 database layer or the R/3 application server layer. If these layers become unavailable, the SAPGUI will display an error message. Inform your SAP systems administrator of the problem and do not continue to use the R/3 system.

What happens when SAP link detects a failure

Depending on what stage of SAP link processing the systems failure occurs, the product will handle the transmitted message as follows:

- On outbound processing:
  1. Rolls back the unit of work (UOW).
  2. Logs the error.
  3. Reports the error to R/3, where it is also logged.
  4. Moves the IDoc message to the Transactional RFC panel (R/3 transaction code `sm58`), where it can be retried after you have corrected the error.
- On inbound processing:
  1. Rolls back the unit of work (UOW).
  2. Logs the error.
  3. Moves the IDoc message back to the inbound queue or to the bad message queue, depending on the type of failure that has occurred.

For more information on error logging, refer to “Using trace and error logging to diagnose problems” on page 69.

Troubleshooting

Handling unrecognizable messages

When the inbound server is waiting for a message to appear on the inbound queue it is looking for the information in a recognizable format; there must be an SAP link header at the front of the message and the information contained in the message must be recognizable to R/3. In other words, the information must be in IDoc format (for more information about the prerequisites for incoming messages see “Message formats” on page 43.

But what happens when a message appears on the queue that the inbound server cannot recognize? It would be inappropriate to leave the 'bad message' on the inbound queue because the inbound server would continually try to retrieve the message and fail (while the queue is set to its default, FIFO), rendering the inbound queue unusable to the inbound server. The inbound server solves this problem by using a *bad message queue* as a repository for these unrecognizable messages, thus clearing the inbound queue to enable processing of subsequent messages.

The inbound server provides a degree of flexibility for the handling of bad messages. An optional parameter in the inbound ini file, `terminateonbadmessage`, controls whether or not the inbound server terminates when an unrecognizable message is encountered. By default, the inbound server will terminate when a bad message is encountered. If this parameter is set to `no`, the server will place the message on the bad message queue, the name of which is also defined as a mandatory parameter in the inbound ini file.

Contents of a bad message

Bad messages are placed onto the bad message queue with a bad message header appended as illustrated below; following on from the description of Message Formats described at “Message formats” on page 43, the complete MQSeries message takes the form:-

MQMD Header	Bad message header	Link header	Application data

Note: The information included in the bad message header is not intended for bad message diagnosis. To perform this task bad message error codes and bad message error types are included in MQSeries link for R/3. Furthermore, the majority of information included in the bad message header appears in hexadecimal format, when viewed with a standard text editor, thus making it difficult to perform any analysis on the header using basic editor tools. Further information can be found about the structure of the bad message header in the supplied `smqc.h` include file.

Bad message diagnosis

When a bad message is generated, the user receives on-screen confirmation that the message has been placed on the bad message queue. Also displayed is the reason code for its classification as a bad message, and the error type. There are two possible error types. Their explanations, and the actions you should take to resolve them are shown in Figure 20.

Figure 20. Error types

Error type	Explanation	Action
1	Either an error was discovered in the SAP link header, or the IDoc data in the message was invalid.	Look up the bad message reason code in Figure 21 and follow the course of action.
2	An error has occurred within the MQSeries processing of the message.	Lookup the reason code in the MQSeries Application Programming Reference manual and follow the course of action specified.

Bad message reason codes

Notes:

1. ƀ is a blank character (ASCII 20).
2. <data> represents the string returned in the explanation message

Figure 21 (Page 1 of 2). Reason codes

Reason code	Explanation	Action
4102	The user exit returned SMQRC_CONVERT_FAIL.	Check the user exit to determine why the conversion failed.
4104	The user exit returned SMQRC_CONVERT_BAD_MESSAGE.	Check the user exit to determine why the message is bad.
4105	The user exit returned SMQRC_TERMINATE_EXIT.	The user exit is terminating. Check the user exit to determine why the conversion failed.
4106	The user exit returned SMQRC_TERMINATE_SERVER.	The server is terminating at the request of the user exit. Check the exit to determine why it requests the server to be terminated.
4107	IDoc has an invalid structure header. IDoc value=<data>. Expected value was “SMQƀ”.	Amend the SAP link header identifier, to “SMQƀ”.

Figure 21 (Page 2 of 2). Reason codes

Reason code	Explanation	Action
4108	IDoc has an invalid structure version. IDoc value=<data>. Expected value was "bbb1dq.. Must be numbers.	Amend the SAP link header version number to "bbb1".
4109	The length of the inbound message is invalid for an IDoc.	Amend IDoc data.
4110	IDoc has an invalid language. IDoc value="<data>"	Amend the SAP link header language (E for English).
4111	IDoc has an invalid client. IDoc value="<data>"	Amend the SAP link header client number.
4112	IDoc has an invalid system number. IDoc value="<data>"	Amend the SAP link header system instance number.
4113	RfcOpen failed. Failed to connect to SAP system for inbound IDoc. Check host and system names.	Check that the R/3 system is contactable and the system instance number and the host name of the remote R/3 system are valid. Amend this information in the SAP link header if necessary.
4114	Failed to get SAP transaction Id. Check Client, User ID, Password, and Language.	Check the remote R/3 connection parameters and the availability of the source and destination R/3 systems.
4115	The attempt to put the message into SAP failed.	Check the format of the IDoc and the ALE partner profile.

Reprocessing bad messages

To allow a bad message to be reprocessed by the inbound server, follow this procedure:

1. Using the reason code given for bad message generation, amend the message according to the instructions in the corresponding 'Action' column.
2. Remove the bad message header from the message.
3. Place the amended bad message back onto the inbound queue. The message will now be processed to the remote R/3 system.

Note: The administrator has two options. The process can either be performed manually, or a program can be written that utilizes information in the bad message header to automatically handle any bad messages.

Using trace and error logging to diagnose problems

There are two types of trace available with SAP link that assist you to undertake a step by step analysis of a problem.

R/3 Remote function call (RFC) trace

When you initialize the server, you can specify whether an RFC trace is to generated. The trace file is named `dev_rfc` and is generated by setting `rfctraceon=y` at outbound or inbound initialization. Once the file is created, any further instances of outbound or inbound errors will append trace information to the same file.

SAP link trace

SAP link trace is provided primarily for the use of IBM service personnel. It traces the functions executed by SAP link. A trace file is generated in the current directory each time a server is started, the file being named SMQnnnnn.TRC, where nnnnn is a unique, non-sequential, five-digit number. You activate this trace by adding `SMQ_TRACE=F` to the list of environment variables on the system where the servers are being run. Contact your systems administrator for platform specific instructions on system environment variables. A TRC trace file is created for each instance of an inbound or outbound server.

SAP link Error logging

As with other MQSeries family products, SAP link generates three types of messages, which, in order of increasing severity are:

- Information messages
- Warning messages
- Error messages

For a complete list of messages, see Appendix C, “Messages and codes” on page 79.

All errors are logged in a set of log files: SMQERR01.LOG, SMQERR02.LOG, and SMQERR03.LOG, which are written to in this order. When one file is filled, the next is started. When all three are filled, the log starts again at SMQERR01.LOG.

When the SAP link servers are started, a log file repository is created as the errors sub-directory. All log files are placed in this repository. The sub-directory resides under the current active directory.

R/3 Error Logging

If R/3 encounters an error while trying to send an IDoc outbound from R/3, the error message is reported back to R/3 and the UOW is rolled back. This roll back occurs whether it is a systems error, an MQSeries error, or an SAP link configuration error.

The error is logged by SAP under R/3 transaction code `sm58`. The administrator can view the error text associated with the message and take the appropriate action to address the error. The message can also be retried from R/3 transaction code `sm58`.

Trace

Security

You can use the security features of MQSeries and SAP link to ensure that only authorized users have access to MQSeries message queues. SAP link maintains all the security features provided by R/3. SAP link can put your IDocs into R/3, but only if you have an authorized user ID and password for that system.

When you start an SAP link server, the server application must be able to connect to the specified queue manager and open the required queues for gets or puts (reading or writing to queues), depending on the type of server and the type of queue.

Startup security

If the MQSeries Object Authority Manager is turned on, any server on that workstation must run under a user ID that is a member of the group `mqm`. Otherwise, the server program cannot connect to the queue manager and the server cannot start.

Message security

Each message carries a user ID in the MQMD header. By default, this is the logon ID of the user in the system originating the message. However, in the outbound ini file, you can specify that the message should take the logon user ID of the outbound R/3 system, as in:

```
usesapuserid=y
```

If you do this, the OAM on the queue manager that owns the queue at the receiving end must be able to accept this user ID. You have to use the appropriate authority command to grant 'put' authority for that user ID. See the *MQSeries System Management Guide* for your platform for details.

Security precautions

To ensure the integrity of SAP passwords, you should ensure that you do not specify SAP passwords in the Message Queuing Options panels for MQSeries. If you do, the outbound server builds the password into the link header of each message it generates.

To prevent passwords being transmitted over the network:

1. Specify the password of the destination R3 system in the inbound ini file, using the `password` parameter.
2. Use the operating system to grant access to any MQSeries or SAP link files only to trusted users. In particular, restrict access to the inbound ini file and any files associated with the inbound message queue.

Alternatively, you could consider encrypting the passwords at the outbound server using user exits, and decrypting them at the inbound server.

You can also write additional security checks, using the security user exits supplied by MQSeries.

Security

Appendix A. Samples

This appendix contains information about the sample files supplied with SAP link. The samples include:

- Initialization (ini) files
- User exits
- MQSC command files

Sample initialization files

File Name	*Purpose*
out.ini	Sample initialization file for use with the outbound server. This file contains a set of optional and required parameters. You can use this file as a basis for your own ini file. For more information about outbound server ini file parameters, see “Outbound server initialization parameters” on page 58. This file is located in the smqso directory.
in.ini	Sample initialization file for use with the inbound server. This file contains a set of optional and required parameters. You can use this file as a basis for your own ini file. For more information about inbound server ini file parameters, see “Inbound server initialization parameters” on page 53. This file is located in the smqsi directory.

Sample user exits

The following sample C source files are supplied:

File Name	*Purpose*
smqesmp1.c	Shows the structure of a user exit. You can use it in either an inbound or outbound server.
smqesmp2.c	Shows the transformation of messages to and from IDoc formats.

See “Sample user exits” on page 41 for more information.

The header files associated with these source files are also supplied in the same samples directory.

Samples

MQSC command file (smqscdef.tst)

The MQSC command file `smqscdef.tst` is located in the `exits` directory. The file contains the following definitions:

SMQ_INBOUND_TRANIDS[1]
Inbound transaction ID queue

This queue stores R/3 transaction IDs for inbound messages.

SMQ_OUTBOUND_TRANIDS[1]
Outbound transaction ID queue

This queue stores R/3 transaction IDs for outbound messages.

SMQ_OUTBOUND_QUEUE
Outbound message queue

The outbound server uses this queue to store outbound messages where they can be retrieved by another application. The server builds these messages from IDocs flowing from an R/3 system. The queue name must match the one specified in the R/3 Message Queuing Options panels.

SMQ_INBOUND_QUEUE
Inbound message queue

The inbound server uses this queue to get inbound messages from MQSeries. The server processes each message, decomposing it into IDocs, which it then passes to the destination R/3 system. The queue name matches the inbound queue name specified in the sample inbound ini file.

SMQ_BADMESSAGE_QUEUE
Bad message queue

The inbound server uses this queue to store any messages that it is unable to convert into valid IDocs. You must ensure that the name of this queue matches the queue name specified in the sample inbound ini file.

Notes:

1. SMQ_OUTBOUND_TRANIDS and SMQ_INBOUND_TRANIDS are the default names for the transaction ID store queues. If you use these names, you can run SAP link without explicitly specifying the transaction queue names in the outbound and inbound ini files.
2. All queues are defined with the default persistence (DEFPSIST) attribute set. All MQSeries messages generated by the link are persistent messages.

Appendix B. Quick reference

This section summarizes the configuration parameters for the inbound and outbound servers. You can use these tables to check your own configurations. Figure 22 summarizes those for inbound configuration; Figure 23 on page 76 summarizes the outbound configuration parameters,

Inbound configuration

Figure 22. Inbound configuration parameters

R/3 Message Queuing Options panel parameters[1]	**Command line parameters on smqsi**[1]	**Outbound initialization file**[1]
-	-i*InitializationFileName*[2]	-
Client[3]	-	client
Language[3]	-	language
User Id[3]	-	userid
Password[3]	-	password
Application Server[3]	-	hostname
System Id[3]	-	sysnbr
Options File (not used)	-	-
-	-	transactionqueue
-	-mQueueManagerName	queuemanager
-	-	terminateonbadmessage
-	-	badmessagequeue
-	-qInboundQueueName	inboundqueue
-	-	exitname
-	-	exitbuffer
-	-	maxconnections
-	-	maxidocsize
-	-	rfctraceon

Notes:

1. The descending precedence of parameters is: Message Queuing Options panel, command line, ini file.
2. This parameter is required for the inbound server.
3. The Message Queuing Options panel parameters are optional. Any parameters you specify here become part of the link header when the MQSeries message is built. These parameters are read by the inbound server when it retrieves the message.

Quick reference

Outbound configuration

Figure 23. Outbound configuration parameters

Message Queuing Options panel parameters in R/3[1]	**Command line parameters on smqso**[1]	**Outbound initialization file**[1]
-	-i*InitializationFileName*[2]	-
Queue manager name[3a]	-	-
-	-m*QueueManagerName*[3b]	queuemanagerehp1.[3b]
Queue name[4]	-	-
User Exit	-	exitname[5]
Exit Buffer[6]	-	exitbuffer[6]
Call user exit[7]	-	-
-	-	rfctraceon
-	-	transactionqueue
-	-	maxidocsize
-	usesapuserid	-
Gateway Service[8] Gateway Host[8] Program Id[9]	-xGatewayService[8] -gGatewayHost[8] -aProgramID[9]	gatewayservice[8] gatewayhost[8] programid[9]

Notes:

1. The descending precedence of parameters is: Message Queuing Options panel, command line, ini file.
2. This parameter is optional for the outbound server.
3. Technically, these queue managers are not necessarily the same:
   a. The queue manager specified in the Message Queuing Options panel in R/3 is the queue manager that owns the outbound queue.
   b. The queue manager specified on the command line or in the ini file specifies the queue manager that owns the transaction ID store queue.

   If not specified, the default queue manager is used.
4. Queue name is a required parameter.
5. For the server to invoke this user exit name, you must also specify an asterisk (*) in the *User Exit* field and check the *Call user exit* box in the Message Queuing Options panel.
6. The use of an exit buffer is optional.
7. See note 5.

8. Gateway service and gateway host, if specified in the RFC destination panel, must match the values actually used on the **smqso** command (taken from the ini file or command line parameters).
9. You define the program ID. The value you specify in the RFC destination panel must match the values actually used on the command (taken from either the ini file or from the command line parameters). Note that the program ID is entered as a character string that matches the name specified in the R/3 RFC destination panel.

Quick reference

Appendix C. Messages and codes

The messages and codes shown in this appendix are generated by SAP link.

Note: Some messages have double asterisks (**) surrounding them; these messages appear on the panel associated with R/3 transaction code `sm58` and the ** is to draw attention to them. They are described in this appendix.

SMQ4101 The SAP link server has started.

Explanation: The SAP link server started normally.

Action: None.

SMQ4102 The SAP link server ended normally.

Explanation: The SAP link server ended normally.

Action: None.

SMQ4103 The SAP link server failed to start. See other messages for details.

Explanation: The SAP link server failed to start. See other messages for a more detailed explanation.

Action: Check other error messages, correct the errors, and restart the server.

SMQ4104 The SAP link server ended abnormally. See other messages for details.

Explanation: The SAP link server ended abnormally with reason code <*reason code*>. See other messages for a more detailed explanation.

Action: Check error messages, correct the errors and re-run the server. If the problem persists, note the errors and reason code, and contact your IBM representative.

SMQ4105 Could not start trace services.

Explanation: The SAP link server could not start trace services. Return code <*ReturnCode*>.

Action: Check that the trace environment variables have been correctly specified.

SMQ4106 Invalid command line arguments. Type server name alone for help.

Explanation: The command line arguments specified are invalid. Type the server name with no parameters for help.

Action: Correct the arguments and restart the server.

SMQ4107 The ini file - <*FileName*> - could not be found.

Explanation: The name of the ini file specified on the command line was not valid or the file does not exist.

Action: Check that the file name and path are correct.

SMQ4108 Invalid ini file parameter at line <*LineNumber*>.

Explanation: An error was detected on line <*LineNumber*> of the ini file.

Action: Correct the error and restart the server.

SMQ4109 The attempt to connect to MQSeries failed. Reason code <*ReasonCode*>.

Explanation: The MQCONN call to queue manager <*QueueManagerName*> failed. The reason code from the MQCONN call was <*ReturnCode*>.

Action: Investigate the MQSeries error code, fix the error, and re-run the server.

SMQ4110 The exit handler detected an invalid state.

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4111 The exit handler has not been initialized.

Explanation: An internal error occurred while processing a user exit because an exit handler component was called before the exit handler was initialized.

Action: Please contact your IBM representative.

SMQ4112 Invalid instance handle passed to exit handler.

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4113 Invalid entry handle passed to exit handler.

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4114 The maximum number of exit instances has been exceeded.

Explanation: A maximum of 256 exits can be run at one time by the exit handler, and this number has been exceeded.

Action: Reduce the number of exits that are run at one time.

SMQ4115 The maximum number of exit entries has been exceeded.

Explanation: A maximum of <*number*> exits can be run at one time by the exit handler, and this number has been exceeded.

Action: Reduce the number of exits that are run at one time.

SMQ4116 Exit <*exit name*> has already been loaded in inbound.

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4117 Exit <*exit name*> has already been loaded in outbound.

Explanation: An exit with the same name has already been loaded in outbound and will therefore not need initializing again.

Action: This is an information message; no action required.

SMQ4118 Invalid command line arguments. See extended text or documentation.

Explanation: The syntax for the start outbound server command is `smqso options`, where the `options` are:

```
-aProgramID
-gGatewayHostName
-xGatewayService
-mQueueManagerName
```

For example:

```
smqso -aBILko.mqserver -ghs0311 -xsapgw01 -mqmgrtest
```

Note: The queue manager name is optional; omit it if you have installed and want to use a default queue manager.

Action: Correct the arguments and restart the server.

SMQ4119 Syntax error at line <*LineNumber*> of ini file.

Explanation: An error was detected on line <*LineNumber*> of the ini file.

Action: Correct the error and restart the server.

SMQ4120 Missing '=' at line <*line number*> of ini file.

Explanation: The lines in the ini file must have the format: parameter=value. An equal sign is missing from a parameter at line <*line number*>.

Action: Correct the error and restart the server.

SMQ4121 The connect to SAP failed. Check gateway host, service and program ID.

Explanation: The attempt to connect to the SAP system failed. Check the gateway service, host name, and program ID for errors. If the error can not be found, enable RFC trace using the option in the ini file, re-run the server and look at the `dev_rfc` trace file.

Action: Correct the errors and restart the server.

MSMQ4122 The attempt to open the default queue manager failed with reason code <*reason code*>.

Explanation: The MQOPEN attempt to open the default queue manager failed with MQSeries reason code <*reason code*>.

Action: Check the MQSeries reason code, correct the error, and restart the server, or specify the name of the queue manager to use.

SMQ4123 The MQINQ call on the default queue manager failed with reason code <*reason code*>.

Explanation: The MQINQ call on the default queue manager failed with MQSeries reason code <*reason code*>.

Action: Check the MQSeries reason code, correct the error, and restart the server, or specify the name of the queue manager to use.

SMQ4124 The attempt to close the default queue manager failed with reason code <*reason code*>.

Explanation: The MQCLOSE attempt on the default queue manager failed with MQSeries reason code <*reason code*>.

Action: Check the MQSeries reason code <*reason code*>, correct the error, and restart the server, or specify the name of the queue manager to use.

SMQ4125 The attempt to open the transaction queue failed with reason code <*reason code*>.

Explanation: The MQOPEN attempt on the transaction queue <*queue name*> failed with MQSeries reason code <*reason code*>.

Action: Check MQSeries reason code <*reason code*>, correct the error, and restart the server.

SMQ4126 The SAP link server ended abnormally. See FFST for details.

Explanation: The SAP link server ended abnormally with reason code <*reason code*>. See FFST for a more detailed explanation.

Action: Please contact your IBM representative.

SMQ4127 Failed to execute exit <*exit name*>.

Explanation: The SAP link server detected an error when attempting to call the exit's execute or return function.

Action: Check that the exit exists in the path specified. Ensure that the exit is not terminating unexpectedly.

SMQ4128 The MQPUT to queue <*queue name*> failed with reason code <*reason code*>.

Explanation: The attempt to put a message to the outbound queue <*queue name*> failed with MQSeries reason code <*reason code*>.

Action: Fix the error, and re-run the server.

SMQ4129 The MQPUT to queue <*queue name*> failed with reason code <*reason code*>.

Explanation: The attempt to put a message to the transaction queue <*queue name*> failed with MQSeries reason code <*reason code*>.

Action: Fix the error, and re-run the server.

SMQ4130 The MQGET from queue <*queue name*> failed with reason code <*reason code*>.

Explanation: The attempt to get a message from the transaction queue <*queue name*> failed with MQSeries reason code <*reason code*>.

Action: Fix the error, and re-run the server.

SMQ4131 The MQCMIT call failed with reason code <*reason code*>.

Explanation: The attempt to commit a transaction failed with MQSeries reason code <*reason code*>.

Action: Fix the error, and re-run the server.

SMQ4132 The MQBACK call failed with reason code <*reason code*>.

Explanation: The attempt to back out a transaction failed with MQSeries reason code <*reason code*>.

Action: Fix the error, and re-run the server.

SMQ4133 Error. <*number1*> IDOCs were expected but only <*number2*> were received.

Explanation: SAP reported that <*number1*> IDOCs were to be sent, but only <*number2*> were received.

Action: Try resending the IDOCs from SAP. If the problem persists, contact your IBM representative.

SMQ4134 An SAP RFC communication error occurred. Turn on RFC trace for details.

Explanation: An SAP RFC communication error occurred.

Action: Use the ini file option to turn on RFC trace, rerun the server, and check the `dev_rfc` trace file for details.

SMQ4135 The memory allocation for the exit anchor block failed.

Explanation: Not enough memory could be allocated for the exit handler.

Action: Try to free up some system resources and restart the server.

SMQ4136 An error occurred while attempting to load the user exit

Explanation: The server failed to load the user exit. The error code from the operating system 'load' function was <*number 1*>

Action: Check that the exit exists.

SMQ4137 An error occurred while attempting to unload the user exit

Explanation: The server could not unload the user exit.

Action: If the error persists, re-run the server with trace enabled and contact your IBM representative.

SMQ4138 The exit handler has run out of handle space

Explanation: The exit handler has attempted to load more exits than are allowed.

Action: Re-run the server. If the problem persists, contact your IBM representative.

SMQ4139 Invalid module handle in exit handler.

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4140 Failed to get pointer to exit anchor block

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4141 Failed to get entry point to exit

Explanation: The server could not call the user exit because the exit entry point (*Initialize*) was not found.

Action: Check that the exit has been written and compiled correctly.

SMQ4142 ** Failed to get tables for IDOCs **

Explanation: An error occurred when attempting to get data from the SAP system.

Action: Re-run the server with RFC trace enabled. Look at the trace file and try to correct the error.

SMQ4143 ** Failed to get RFC Attributes **

Explanation: An error occurred when attempting to get data from the SAP system.

Action: Re-run the server with RFC trace enabled. Look at the trace file and try to correct the error.

SMQ4144 ** Failed to get destination data **

Explanation: An error occurred when attempting to get data about the RFC Destination from the SAP system.

Action: Re-run the server with RFC trace enabled. Look at the trace file and attempt to correct the error.

SMQ4145 ** Failed to initialize exit handler **

Explanation: An internal error occurred while processing a user exit.

Action: Please contact your IBM representative.

SMQ4146 ** Failed to copy IDOCs into memory buffer **

Explanation: An error occurred when attempting to write data into memory, check other error messages for more details.

Action: Fix the error and resend the IDOCs.

SMQ4147 ** The RfcSendData failed **

Explanation: An error occurred when attempting to send data to SAP.

Action: Re-run the server with RFC trace enabled for more information.

SMQ4148 ** User exit <*exit name*> failed. **

Explanation: An error occurred while processing a user exit.

Action: See other error messages for more information.

SMQ4149 ** MQPUT failed - Reason <*reason code*> **

Explanation: The MQPUT failed with MQSeries reason code <*reason code*>.

Action: Please fix the error and send the IDOC again.

SMQ4150 ** Failed to read IDOCs from ITABs **

Explanation: An error occurred while reading the IDOCs from SAP.

Action: Run the server with RFC trace, resend the IDOCs and check the trace file for more details about the error.

SMQ4151 ** Terminating outbound server **

Explanation: The IDOC cannot be processed because the outbound server is ending.

Action: See other messages to find out why the server is ending.

SMQ4152 No inbound queue name was specified.

Explanation: An inbound MQSeries queue name must be specified on the command line or in the ini file.

Action: Restart the server specifying an inbound queue name.

SMQ4153 No Bad Message queue name was specified.

Explanation: A Bad Message queue name must be specified in the ini file if the `terminateonbadmessage` parameter is set to N.

Action: Restart the server specifying the name of the bad message queue.

SMQ4154 The attempt to open the inbound queue failed with reason code <*reason code*>.

Explanation: The MQOPEN attempt on the inbound queue <*queue name*> failed with MQSeries reason code <*reason code*>.

Action: Check MQSeries reason code <*reason code*>, correct the error, and restart the server.

SMQ4155 The attempt to open the bad message queue failed with reason code <*reason code*>.

Explanation: The MQOPEN attempt on the bad message queue <*queue name*> failed with MQSeries reason code <*reason code*>.

Action: Check MQSeries reason code <*reason code*>, correct the error, and restart the server.

SMQ4156 The MQGET from the inbound queue failed with error code <*error code*>.

Explanation: The attempt to get a message from the inbound message queue failed with reason code <*reason code*>. The inbound server will end.

Action: Fix the error and restart the inbound server.

SMQ4157 The server received a termination request from the user exit.

Explanation: An exit program that was called by the server returned SMQRC_TERMINATE_SERVER, so the server will be terminated.

Action: None.

SMQ4158 Failed to connect to SAP system for inbound IDOC. Check host and system name.

Explanation: The RfcOpen call failed for an IDOC on the inbound queue using host name <*host name*> and system number <*number*>. Check that the host name and system number are valid. These are taken from the IDOC or from the initialization file.

Action:

SMQ4159 The client is missing from logon information in IDOC or ini file.

Explanation: There is no client specified in the logon information for the IDOC and no default client has been specified in the ini file.

Action: Ensure that the RFC destination is set up correctly in SAP (transaction SM59) or supply a default client in the inbound ini file.

SMQ4160 Language is missing from logon information in IDOC or ini file.

Explanation: There is no Language specified in the logon information for the IDOC and no default language has been specified in the ini file.

Action: Ensure that the RFC destination is set up correctly in SAP (transaction SM59) or supply a default language in the inbound ini file.

SMQ4161 Host name missing from logon information in IDOC or ini file.

Explanation: There is no Host name specified in the logon information for the IDOC and no default host has been specified in the ini file.

Action: Ensure that the RFC destination is set up correctly in SAP (transaction SM59) or supply a default host name in the inbound ini file.

SMQ4162 User ID missing from logon information in IDOC or ini file.

Explanation: There is no User ID specified in the logon information for the IDOC and no default user has been specified in the ini file.

Action: Ensure that the RFC destination is set up correctly in SAP (transaction SM59) or supply a default user ID in the inbound ini file.

SMQ4163 Password missing from logon information in IDOC or ini file.

Explanation: There is no password specified in the logon information for the IDOC and no default password has been specified in the ini file.

Action: Ensure that the RFC destination is set up correctly in SAP (transaction SM59) or supply a default password in the inbound ini file.

SMQ4164 System number missing from logon information in IDOC or ini file.

Explanation: There is no System number specified in the logon information for the IDOC and no default system number has been specified in the ini file.

Action: Ensure that the RFC destination is set up correctly in SAP (transaction SM59) or supply a default system number in the inbound ini file.

SMQ4165 Warning on MQGET from inbound queue. Reason code <*reason code***>.**

Explanation: The attempt to get a message from the inbound message queue returned a warning. The warning code was <*number*>. The message will be passed to the user exit, if one has been specified. Otherwise, the message is passed to the bad message queue. If no bad message queue was specified, the message is left on the inbound queue, and the server terminates.

Action: None.

SMQ4166 IDOC has an invalid structure header. IDOC value=<*value1***> Expected value=<***value2***>.**

Explanation: The value of the structure ID field in the IDOC header structure contains an invalid value.

Action: Ensure that the IDOC message starts with a valid version of the SAP link header structure (MQSAPH).

SMQ4167 IDOC has an invalid structure version. IDOC value=<*value1***>. Expected value=<***value2***>.**

Explanation: The value of the version field in the IDOC header structure contains an invalid value.

Action: Ensure that the IDOC message starts with a valid version of the SAP link header structure (MQSAPH).

SMQ4168 IDOC has an invalid system number. IDOC value=<*value1***>.**

Explanation: The value of the system number field in the IDOC header structure contains an invalid system number. The system number specified in the header structure should contain only digits or blanks.

Action: Ensure that the IDOC message starts with a valid version of the SAP link header structure (MQSAPH). If the IDOC was generated by SAP, check that the system number in the RfcDestination is valid.

SMQ4169 IDOC has an invalid client. IDOC value=<*value1*>.

Explanation: The value of the client field in the IDOC header structure contains an invalid client. The client specified in the header structure should contain only digits.

Action: Ensure that the IDOC message starts with a valid version of the SAP link header structure (MQSAPH). If the IDOC was generated by SAP, check that the client in the RfcDestination is valid.

SMQ4170 IDOC has an invalid language. IDOC value=<*value1*>.

Explanation: The value of the language field in the IDOC header structure contains an invalid language. The language specified in the header structure should contain only a character or a blank.

Action: Ensure that the IDOC message starts with a valid version of the SAP link header structure (MQSAPH). If the IDOC was generated by SAP, check that the language in the RfcDestination is valid.

SMQ4171 Failed to get SAP transaction ID. Check Client, User ID, Password and Language.

Explanation: The request for a transaction Id from SAP failed. This may be because the logon information in the IDOC or the ini file is not valid or because the connection to R/3 has been lost. To get more information, turn on the RFC trace option in the ini file, re-run the server and check the RFC trace (dev_rfc) file.

Action: None.

SMQ4172 The length of the inbound message is invalid for an IDOC.

Explanation: The inbound message data length is not compatible with the IDOC format. The message should be comprised of: The SAP link header structure; followed by an IDOC control structure and IDOC data records for each IDOC in the message file.

Action: Ensure that the IDOCs have the correct format.

SMQ4173 The server is terminating because an invalid message was received.

Explanation: See previous messages to find out why the message is not valid.

Action: To prevent the server from terminating when bad messages are received, set the `terminateonbadmessage` parameter in the ini file to N. None.

SMQ4174 Inbound IDOC number <*number*> received from queue.

Explanation: None

Action: None.

SMQ4175 Outbound IDOC number <*number*> received.

Explanation: None

Action: None.

SMQ4176 The value is too long. Max. length is 48. (Error in value: <*value*>)

Explanation: The command line argument is too long.

Action: Change the value specified on the command line and re-run the server.

SMQ4177 Options must begin with '-' and contain no spaces. (Error in: <*option*>)

Explanation: An invalid option was specified on the command line.

Action: Change the value specified on the command line and re-run the server.

SMQ4178 No initialization file name was specified. This is required.

Explanation: You must specify the name of the initialization file.

Action: Specify the initialization file on the command line and re-run the server.

SMQ4179 Invalid option. Valid options are: '-q', '-m', and '-i'. (Error in:<*option*>)

Explanation: The valid command line options are:

-i*IniFileName* (required)
-q*QueueName*
-m*QueueManagerName*

Action: Change the values specified on the command line and re-run the server.

SMQ4180 Invalid option. Valid options are: -a,-g,-x,-m and -i. (Error in:<*option*>)

Explanation: The valid command line options are:

-i*IniFileName*
-a*ProgramID*
-g*GatewayHost*
-x*GatewayService*
-m*QueueManagerName*

Action: Change the values specified on the command line and re-run the server.

SMQ4181 Options -a, -g, and -x must be specified if there is no ini file.

Explanation: You must specify -a, -g, and -x if there is no ini file.

Action: Change the value specified on the command line and re-run the server.

SMQ4182 SAP link FFST in progress.

Explanation: An internal error has occurred and data is being dumped to an FDC file.

Action: Save the error log and FDC dump files and contact your IBM representative.

SMQ4183 An IDOC was not put to the message queue, see the previous errors for the reason.

Explanation: None.

Action: None.

SMQ4184 Execution of smqUserExitReturn failed in user exit <*user exit name*>

Explanation: The IDOC failed because an error occurred when the return function of the user exit was called.

Action: Investigate the cause of the failure, fix the error, and resend the IDOC.

SMQ4185 User exit <*UserExitName*> failed to convert message. Transaction will be rolled back.

Explanation: The user exit returned SMQ_CONVERT_FAIL. It was not able to convert the message.

Action: Investigate the user exit code and the IDOC conversion functions.

SMQ4186 An invalid result has been received from the user exit Return function.

Explanation: The user exit returned an invalid result for this call. Valid result codes are: SMQRC_OK, SMQRC_TERMINATE_SERVER, or, SMQRC_TERMINATE_EXIT.

Action: Change to user exit Return function so that it returns valid result codes.

SMQ4187 The attempt to put the message into SAP failed.

Explanation: The call to RfcIndirectCall for INBOUND_IDOC_PROCESS failed.

Action: For more information re-run the server with trace and RFC trace enabled.

SMQ4188 Reason code <*reason code*> when opening queue <*queue name*>

Explanation: The MQOPEN call to queue <*queue name*> failed. The return code from the MQOPEN call was <*reason code*>.

Action: Investigate the MQSeries error code, fix the error and retry the transaction.

SMQ4189 The Inbound queue must be different to the Bad Message queue.

Explanation: You have specified the same MQSeries queue for inbound messages and bad messages. This is not allowed.

Action: Change the name of one of the queues and rerun the server. transaction.

SMQ4190 The IDOC is larger than the size specified in the ini file.

Explanation: The IDOC or IDOC package is larger than the size specified in the maxidocsize parameter in the ini file..

Action: Increase the maxidocsize parameter in the ini file, or send IDOCs in smaller packages.

SMQ4191 An message was put to the bad message queue. Bad message type <*number*>, reason <*reason code*>.

Explanation: The message was not in a valid IDOC format. The bad message type is <*number*> and the bad message reason is <*reason code*>.

Action: Check the bad message reason code in the bad message header of the message. Attempt to correct the error and send the message again.

SMQ4192 A data conversion problem occurred on the MQGET. Attempting to process message.

Explanation: A warning was issued because the message needs codepage conversion, but the message is either not in MQSTR format, or a user-defined data- conversion exit call failed.

Action: Ensure that incoming messages from machines with a different code page, are in MQSTR format or that there is a user exit defined to convert messages in other formats.

SMQ4193 The exit requested termination. If a transaction was in progress it will be aborted.

Explanation: An exit program that was called by the server returned SMQRC_TERMINATE_EXIT so the exit will be terminated, and any transactions in progress will be aborted.

Action: None.

SMQ4194 An invalid result <*result code*> has been received from the user exit Execute function.

Explanation: The user exit returned an invalid result for this call. Valid result codes are: SMQRC_OK, SMQRC_TERMINATE_SERVER, SMQRC_TERMINATE_EXIT, SMQRC_CONVERT_OK, SMQRC_CONVERT_FAIL, SMQRC_CONVERT_NOT_NEEDED, or, SMQRC_CONVERT_BADMESSAGE.

Action: Change to user exit Return function so that it returns valid result codes.

SMQ4195 An invalid result <result code> has been received from the user exit Initialize function.

Explanation: The user exit returned an invalid result for this call. Valid result codes are: SMQRC_OK, SMQRC_TERMINATE_SERVER, or, SMQRC_TERMINATE_EXIT.

Action: Change to user exit Return function so that it returns valid result codes.

SMQ4196 The MQPUT to queue <queue name> failed with reason code <reason code>.

Explanation: The attempt to put a message to the Bad Message queue failed with MQSeries reason code <reason code>.

Action: Fix the error and re-run the server.

SMQ4197 <text>

Explanation: The above text details an error that occurred in an R/3 RFC function.

Action: None. .

SMQ4198 The maximum number of connections to R/3 has been exceeded.

Explanation: The inbound server has attempted to make more connections to R/3 than are allowed.

Action: Increase the maxconnections parameter in the ini file if it was specified. If it was not specified, the default of 256 was used. Specify a higher value than this in the ini file.

Appendix D. Notices

The following paragraph does not apply to any country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used.

Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction other products, except those expressly designated by IBM, are the responsibility of the user.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact Laboratory Counsel, MP151, IBM United Kingdom Laboratories, Hursley Park, Winchester, Hampshire, England SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Notices

Appendix E. Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	AIX/6000	AS/400
BookManager	CICS	IBM
MQ	MQSeries	MVS/ESA
OS/2	OS/400	

The following terms are registered trademarks of SAP AG:

SAP	R/2	R/3

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Trademarks

Glossary of terms and abbreviations

This glossary describes terms used in this book and words used with other than their everyday meaning. In some cases, a definition may not be the only one applicable to a term, but it gives the particular sense in which the word is used in this book.

If you do not find the term you are looking for, see the Index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

ABAP/4. The 4th generation SAP programming language.

application server. An R/3 term for a system on which one or more applications are running. Applications may be R/3 applications.

aRFC. Asynchronous remote function call (an R/3 term)

bad message queue. A queue used by the inbound server to store incoming MQSeries messages when the messages cannot be decomposed into valid IDocs.

exit handler. A subcomponent of the inbound (or outbound) server that calls a user-written program, as specified in the ini file for that server. See also *user exit.*

First Failure Support Technology (FFST). Used by some members of the MQSeries family of products to detect and report software problems.

gateway server. An R/3 gateway through which R/3 applications and SAP link communicate.

local queue. An MQSeries queue that belongs to the local queue manager. A local queue can contain one or more MQSeries messages waiting to be processed.

IDoc. Intermediate document.

A data container used by R/3 applications to send and receive information. SAP link works only with IDocs.

inbound message queue. An MQSeries queue from which the inbound server takes messages for processing and passes them to the R/3 system.

inbound server. The component of SAP link that receives MQSeries messages from a specified queue and converts them to IDocs that can be used by R/3.

inbound transaction id queue. A queue used by the inbound server to store transaction IDs associated with an SAP transaction. These IDs are used for backing out transactions that occur within a syncpoint, if a unit of work needs to be rolled back.

ini file. See *initialization file*.

initialization file. A file from which an SAP link server (inbound or outbound) takes data when it is started.

message. In message queuing applications, a communication sent from one application or program to another.

outbound message queue. An MQSeries queue on which the outbound server puts messages containing IDoc data originating from the R/3 system.

outbound server. The component of SAP link that receives one or more IDocs from an R/3 system and converts them into an MQSeries message, which it puts on a specified MQSeries queue.

outbound transaction id queue. A queue used by the outbound server to store transaction IDs associated with an R/3 transaction. These IDs are used for backing out transactions that occur within a syncpoint, if a unit of work needs to be rolled back.

remote queue. An MQSeries object, belonging to the local queue manager, that identifies a local queue on another queue manager.

RFC. remote function call (R/3 term)

SAP link. An abbreviation for *MQSeries link for R/3*.

SAP link commands. A set of commands that you can invoke from the command line to control SAP link.

SM59. An R/3 transaction code that invokes the destinations panels in R/3 so that you can configure R/3 destinations and MQSeries queuing options for the SAP link.

transaction queue. See *inbound transaction id queue* and *outbound transaction id queue.*

translator tools. Programs that can translate IDocs from one format to another. These programs can be called by a user exit.

Glossary

user exit. A user written program that processes data being transferred through SAP link. The user exit is called by the exit handler on either an inbound or an outbound server.

Index

Index

N

O

P

Q

R

S

Index

Program Number: 5765-B66

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC33-1934-00