Copyright

© Copyright 2001 SAP AG. All rights reserved.

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

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

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

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

ORACLE® is a registered trademark of ORACLE Corporation.

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Caution Icon" /></td>
<td>Caution</td>
</tr>
<tr>
<td><img src="image" alt="Example Icon" /></td>
<td>Example</td>
</tr>
<tr>
<td><img src="image" alt="Note Icon" /></td>
<td>Note</td>
</tr>
<tr>
<td><img src="image" alt="Recommendation Icon" /></td>
<td>Recommendation</td>
</tr>
<tr>
<td><img src="image" alt="Syntax Icon" /></td>
<td>Syntax</td>
</tr>
</tbody>
</table>
Inhalt

SAP Communication: CPI-C Programming (BC-CST-GW).......................... 8
SAP Communication .......................................................................... 9
SAP Interfaces .................................................................................. 10
  Communications Basis CPI-C ....................................................... 11
  Remote Function Call (RFC) ......................................................... 13
  Queue Application Programming Interface (Q-API) ......................... 14
Communication in an IBM Host Environment (SNA) ......................... 15
Communication in a BS2000 Host Environment .................................. 16
Communication in a TCP/IP Environment ........................................... 17
Programming under Various Constellations ...................................... 18
Communication Between R/3 Systems .............................................. 19
  Communication Between R/3 and R/2 (MVS/VSE) .......................... 20
    From R/3 to R/2 ...................................................................... 21
    From R/2 to R/3 ...................................................................... 23
  Communication Between R/3 and R/2 (BS2000) .............................. 25
    From R/3 to R/2 ...................................................................... 26
    From R/2 to R/3 ...................................................................... 28
Communication Between R/3 and an Externally Registered Program .... 30
  From R/3 to an External Program (registered program) .................... 31
  From an External Program to R/3 ............................................... 36
    Registered Program ............................................................... 39
Communication Between R/2 Systems .............................................. 40
  Communication Between R/2 and an Externally Registered Program 41
    From R/2 to an External Program (registered program) ............... 42
    From an External Program to R/2 ......................................... 46
Communication Between C Programs .............................................. 49
CPI-C Implementation in ABAP ......................................................... 51
Agreements Between Sender and Recipient ...................................... 53
  Setting the Send/Receive Mode .................................................. 54
  Synchronization ....................................................................... 55
  Setting the Size of Transfer Units .............................................. 56
  Selecting Data Types and Structures .......................................... 57
  Data Conversion ........................................................................ 58
  Remotely Attachable ABAP Program .......................................... 59
  SAP Logon Protocol for External CPI-C Programs ......................... 62
  Establishing a Connection via an ABAP Program ......................... 65
CPI-C Implementation in ABAP ......................................................... 67
  COMMUNICATION INIT: Initialization ........................................ 68
  COMMUNICATION ALLOCATE: Set up connection ...................... 70
  COMMUNICATION ACCEPT: Accept connection ......................... 72
  COMMUNICATION SEND: Send Data .......................................... 74
  COMMUNICATION RECEIVE: Receive data ................................ 76
  COMMUNICATION DEALLOCATE: Close the connection ............... 78
  Return Codes .......................................................................... 79
CPI-C Interface in C ........................................................................ 82
# CPI-C Development Libraries

- Libraries Based on LU6.2: cpicslib ................................................................. 84
- Libraries Based on TCP/IP: cpictlb ................................................................. 86
- Linking an SAP Development Library ............................................................ 87

## Implemented CPI-C Function Calls

- CPI-C Starter Set ............................................................................................. 89
  - CMINIT ............................................................................................................. 91
  - CMACCP ........................................................................................................... 92
  - CMALLC .......................................................................................................... 94
  - CMSEND ......................................................................................................... 95
  - CMRCV .......................................................................................................... 96
  - CMDEAL ......................................................................................................... 98
- Advanced Function Calls ................................................................................. 99
  - CMCFM .......................................................................................................... 100
  - CMCFMD ........................................................................................................ 101
  - CMCNVO ....................................................................................................... 102
  - CMCVNI ........................................................................................................ 103
  - CMSCSP ........................................................................................................ 104
  - CMSCST ........................................................................................................ 105
  - CMSCSU ........................................................................................................ 106
  - CMSPLN ........................................................................................................ 107
  - CMSTPN ........................................................................................................ 108
  - CMSSL .......................................................................................................... 109
- SAP-Specific CPI-C Functions ......................................................................... 110
  - SAP_CMINIT ................................................................................................... 111
  - SAP_CMACCP ............................................................................................... 112
  - SAP_CMPERR .............................................................................................. 113
  - SAP_CMLOGON ............................................................................................. 114
  - SAP_CMCERR .............................................................................................. 116
  - SAP_CMLOADCONVTAB .............................................................................. 117
  - SAP_CMMODCONVTAB ............................................................................... 118
  - SAP_CMTIMEOUT ........................................................................................ 119
  - SAP_CMGWSERV ......................................................................................... 120
  - SAP_CMGWHOST ......................................................................................... 121
  - SAP_CMHANDLE ......................................................................................... 120
  - SAP_CMPERR .............................................................................................. 112
  - SAP_CMACCP ............................................................................................... 113
  - SAP_CMINIT ................................................................................................... 114
- Functions for Registered CPI-C Programs .................................................... 123
  - SAP_CMCANCREGTP .................................................................................. 127
- SNC Function Calls ......................................................................................... 128

## Define Variables for Host Types

- Asynchronous Data Transfer With Q-API ...................................................... 132
- Data Transfer ................................................................................................. 133
- Queues .......................................................................................................... 135
- Queue Attributes ......................................................................................... 136
- Queue Element ............................................................................................ 137
- Queue Unit ................................................................................................. 138
- Queue Interface in the R/3 System ............................................................... 139
 QUEUE_OPEN .......................................................................................................................... 140
 QUEUE_PUT ............................................................................................................................ 142
 QUEUE_GET ........................................................................................................................... 143
 QUEUE_CLOSE ....................................................................................................................... 144
 QUEUE_ERASE ...................................................................................................................... 145
 QUEUE_DELETE ................................................................................................................... 146
 QUEUE_SCHEDULE ............................................................................................................. 147

 Queue Interfaces in the R/2 System ......................................................................................... 148
 Queue Interface for Release 5.0 ............................................................................................ 149
   ABAP Key Words ................................................................................................................ 150
   Queue Parameters ............................................................................................................. 151
   ABAP Statements ............................................................................................................. 155
 Transfer in BS2000 ............................................................................................................. 156
   Asynchronous Driver Communication .............................................................................. 157
     Recipient is not an SAP System ................................................................................... 160
     Recipient is not an SAP System ................................................................................... 161
   Queue Transfer Without Buffering ................................................................................. 162
 Notes on Installation .......................................................................................................... 163
 Extensions ............................................................................................................................. 164
 SAP ACCOUNTING Interface SAPSTEC (as of Release 4.3J, 4.4C and 5.0A) ......................... 166

 Queue Interface for C Programs: RFC to R/3 ........................................................................ 171
 Using SAP Test Programs ...................................................................................................... 174
 Available SAP Test Programs .............................................................................................. 175
 Specifying Program Parameters .......................................................................................... 177
 Requirements for Starting an External Partner Program ...................................................... 178
 Testing Connections ............................................................................................................ 179
   Calling Program: ABAP Program in R/3 ....................................................................... 180
   Calling Program: ABAP Program in R/2 ....................................................................... 181
   Calling Program: Program Written in C ....................................................................... 183

 Error Analysis ....................................................................................................................... 185
 Function SAP_CMPERR ....................................................................................................... 186
 Error Analysis Under OS/2 ................................................................................................. 187
 Error Analysis Under UNIX and WindowsNT ................................................................. 189
 Error Messages of the SAP Transfer Program .................................................................... 191
 Special Features on R/2 Hosts ............................................................................................. 195
 BS2000 R/2 Host: UTM-UTM Connection .......................................................................... 196
   Initiator: R/2 System ......................................................................................................... 199
   Initiator: Non-SAP system .............................................................................................. 200
 MVS/VSE R/2 Host: CICS and IMS ...................................................................................... 201
   CICS Special Features ................................................................................................... 202
   IMS Special Features ................................................................................................... 203
 Sample Programs .................................................................................................................. 204
 Sample Program for R/3 ....................................................................................................... 205
 Sample Programs for R/2 Release 5.0 .................................................................................. 208
   Program RSAPPQ10 ........................................................................................................ 209
   Program RSAPPQ20 ........................................................................................................ 212
   Program RSAPPQ30 ........................................................................................................ 214
 APC and APQ Headers DSECT for Assembler Programs ...................................................... 216
SAP Communication: CPI-C Programming (BC-CST-GW)

In this documentation, you will learn how to implement program-to-program communication between SAP Systems (R/3 or R/2) and with external programs and systems using SAP’s CPI-C interfaces.

CPI-C programming is only required in the following cases:

- Your program communicates with systems that do not support RFC (R/2 before Release 5.0D).
- You want to define your own protocol on the basis of CPI-C.
- You want to change an existing program which contains CPI-C calls.

The Remote Function Call interface further simplifies the implementation of communication. RFC is an SAP interface based on CPI-C. For more details on RFC, refer to the documentation Remote Communications [Extern].

For details on configuration, please refer to the documentation BC - SAP Communication Configuration [Extern].

The following topics are discussed in this documentation:

- SAP Communication [Seite 9]
- Programming under Various Constellations [Seite 18]
- CPI-C Implementation in ABAP [Seite 51]
- CPI-C Interface in C [Seite 82]
- Asynchronous Data Transfer With Q-API [Seite 132]
- Using SAP Test Programs [Seite 174]
- Error Analysis [Seite 185]
- Special Features on R/2 Hosts [Seite 195]
- Example Programs [Seite 204]
- Conversion Tables EBCDIC from/to ASCII [Seite 219]
- Special Features on R/2 Hosts [Seite 195]
SAP Communication

This chapter explains basic terms and discusses communication in various forms of the SAP environment.

- **SAP Interfaces [Seite 10]**
- **Communication in an IBM Host Environment (SNA) [Seite 15]**
- **Communication in a BS2000 Host Environment [Seite 16]**
- **Communication in a TCP/IP Environment [Seite 17]**
SAP Interfaces

Purpose
SAP program interfaces simplify and standardize communication between different systems and/or programs.

SAP communication interfaces exist at various levels. They are described in the following. The main subject of this documentation is the SAP communication interface CPI-C.

The following SAP communications interfaces are available:

- Communications Basis CPI-C [Seite 11]
- Remote Function Call (RFC) [Seite 13]
- Queue Application Programming Interface (Q-API) [Seite 14]
Communications Basis CPI-C

Definition

The Common Programming Interface - Communications (CPI-C) is a standard call interface for applications, which perform direct program-to-program communication.

CPI-C was first defined as a standardized communications interface by IBM in 1987, as part of the SAA standard.

CPI-C was modified by X/Open to include additional functions. SAP's CPI-C implementations support the X/Open Developers' Specification - CPI-C.

The main advantage of CPI-C is the easy portability of programs to various system platforms made possible by the common interface.

Use

The CPI-C communications interface essentially fulfills the following requirements of program-to-program communication:

- Communication setup
- Data exchange
- Data conversion (ASCII ↔ EBCDIC)
- Communication control
- Communication close

Structure

The CPI-C interface can be split into two function groups. This division does not, however, imply limitations in respect of possibilities to use and combine the functions. The function groups are there merely to guide the user:

- CPI-C Starter Set
- Advanced Function Calls

CPI-C Starter Set

These basic functions represent the minimum range of functions shared by two partner programs:

- Establishing a connection
- Data exchange
- Closing a connection

As these are the basic functions of a communication protocol, it is possible to reproduce the CPI-C Starter Set on protocols other than LU6.2.

The SAP CPI-C development library cpictlib is an example of mapping to TCP/IP.

Advanced Function Calls

These advanced functions essentially cover the following task areas:
Communications Basis CPI-C

- Data Conversion
- Synchronization and control
- Changes in communication characteristics
- Checking of communication characteristics
- Security functions

Integration

The CPI-C interface is available for both C and ABAP programs.

CPI-C Development Libraries

Function call interfaces for the C language.

These platform-specific and protocol-dependent libraries implement a series of function calls of the CPI-C communications interface. They also include SAP-specific function calls.

Function calls from the corresponding library enable communication between an external program and an ABAP program or an external program.

See also CPI-C interface in C [Seite 82].

CPI-C Interface in ABAP

This is the function call interface for ABAP.

It implements some function calls of the CPI-C interface.

This interface allows an ABAP program to communicate with an ABAP program from another SAP system (R/2 or R/3) or with a non-SAP program.

Further details are available in the section on CPI-C Implementation in ABAP [Seite 51].

A detailed description of the CPI-C interface is provided in the documentation on BC - SAP Communication: CPI-C Programming [Extern].
Remote Function Call (RFC)

Definition
This interface is logically above CPI-C. It simplifies the implementation of communication processes by relieving the programmer of the task of writing his own communication routines.

Use
The RFC interface enables function calls between two SAP systems (R/3 or R/2), or between an SAP system and an external one. The RFC library functions support the C programming language and Visual Basic (on Windows platforms).

In the case of asynchronous RFC, calls are also transmitted to remote systems when the target system is not active or momentarily cannot be reached (analog to Q-API).

For more details on this interface, refer to the following documentation:

Remote Communications [Extern]
Queue Application Programming Interface (Q-API)

Definition

Q-API is an interface for buffered data transfer. Data is transferred to the partner system using CPI-C.

This is a set of functions, which places the data temporarily in a database queue, to be processed later by a program running asynchronously.

Use

This SAP interface allows asynchronous data exchange between two systems (R/3, R/2 or an non-SAP system).

As of R/3 Release 3.0 you can use the transactional RFC for buffered data transfer.

The transactional RFC is not supported in R/2.
Communication in an IBM Host Environment (SNA)

Definition
A logical connection (Session [Extern] Error! No bookmark name given.) between two LUs (Logical Unit [Extern] Error! No bookmark name given.) is necessary for communication in a homogeneous SNA network. The SNA protocol LU6.2 is used when two application programs are to communicate via a session of this type. Active communication is known as a conversation [Extern].

A conversation between programs, which use different interfaces for the LU6.2 function calls, is possible.

Examples of such interfaces are:
- CPI-C, APPC [Extern], EXEC CICS...

One of the most important characteristics of the LU6.2 [Extern] protocol is that a transaction program [Extern] can call up a partner program on another system (Attach function). This allows connections to be set up dynamically and event-orientated data exchange between the two partners.

Integration
SAP offers a platform-specific development library for workstations which communicate with an R/2 SNA system.

A platform-specific SNA communications subsystem must be installed and properly configured on the workstation:
- SNAplusLink (HP)
- SNA Server (IBM)
- Transit (SNI)
- SNA Server (WindowsNT)

Communication between the workstation and host generally takes place via the SAP Gateway [Extern] (CPI-C Handler).

For information on hardware and software supported, refer to the following brochure: SAP - Supported Network Products.
Communication in a BS2000 Host Environment

Definition

As in the SNA world, a session [Extern] between two network users is used as a medium for data interchange.

In a BS2000 environment, program-to-program communication is possible with the following constellations:

- **UTM on both partner systems**
  
  A conversation [Extern] between two programs is based on UTM-D.

- **DCAM with the SAP Gateway on the BS2000 host**
  
  In addition to an R/3 System, any external system can communicate with the R/2 System via the SAP Gateway [Extern] and DCAM, providing the following requirements are met:
  
  - TCP/IP support
  - SAP communication interfaces (function libraries cpictlib or librfc)

  The SAP Gateway allows communication both with an UTM R/2 System and with a DCAM R/2 System.

For information on hardware and software supported, refer to the following brochure: *SAP - Supported Network Products*.

Detailed documentation on the SAP Gateway for BS2000 is supplied with the gateway, in the SAPGW.README file.
Communication in a TCP/IP Environment

In the following constellations, program-to-program communication is based on the TCP/IP transport protocol:

- R/3 ↔ R/3
- R/3 ↔ Non-SAP Program
  
  For non-SAP programs, SAP provides the platform-specific development library `cpictlib`.

- R/3 (or non-SAP program) ↔ R/2 in BS2000

  The SAP Gateway runs under DCAM (from V11) with TCP/IP and the Socket interface on the BS2000 host.

In all of these constellations, the SAP Gateway [Extern] (CPI-C Handler) is required.

For information on hardware and software supported, refer to the following brochure: SAP-Supported Network Products
Programming under Various Constellations

Various constellations are possible for communication between programs within the SAP world (R/2 and R/3), and between SAP programs and external programs.

Communication is based on various protocols, depending on constellation:

- SNA-LU6.2
- TCP/IP

If the R/2 host is a BS2000 System, the SAP Gateway runs under DCAM in BS2000 and communicates with R/3 or an external program on the basis of TCP/IP.

The SAP Gateway is always necessary for communication via one of the interfaces implemented by SAP.

The individual constellations are described in the following.

- R/3 ↔ R/3 [Seite 19]
- R/3 ↔ R/2 (MVS/VSE) [Seite 20]
- R/3 ↔ R/2 (BS2000) [Seite 25]
- R/3 ↔ Non-SAP Program [Seite 30]
- R/2 ↔ R/2 [Seite 40]
- R/2 ↔ Non-SAP Program [Seite 41]
- C Program ↔ C Program [Seite 49]
Communication Between R/3 Systems

Purpose
An ABAP program in an R/3 System uses an RFC or CPI-C call to start an ABAP program in another R/3 System, and exchanges data with this program.

With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - Choose the target system using transaction SM59.
  - The called function must be an ABAP function module, which has the “remote” ID in the function library.

For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

- Using the CPI-C interface
  - The calling ABAP program must build the data for the logon to the target system, convert it to EBCDIC and receive the response (possibly a denial) from the target system.
  - The side info table TXCOM must be configured in the calling R/3 System. For this, use transaction code SM54.

For details on the configuration of TXCOM, refer to the relevant topic in the following documentation:

BC SAP Communication: Configuration [Extern]

- The target ABAP program must contain the form routine specified in the connect data.

For details on CPI-C calls, refer to the topic CPI-C Implementation in ABAP [Seite 51].
Communication Between R/3 and R/2 (MVS/VSE)

Purpose
The following topics provide an overview about the prerequisites for the partner systems.

💡

As of R/2 Release 5.0D, you can also use an RFC call instead of a CPI-C call.

The following limitation applies to an MVS/VSE host:

- CICS only as the DC system (at present)
- IMS as of Version 4.1 for complete LU6.2 support

These prerequisites depend on whether the calling system is R/3 or R/2:

- From R/2 to R/3 [Seite 23] [Seite 21]
- From R/3 to R/2 [Seite 21] [Seite 23]
From R/3 to R/2

Purpose

An ABAP program in an R/3 System uses an RFC or CPI-C call to start an ABAP program in an R/2 System on an MVS/VSE host, and exchanges data with this program.

The SAP Gateway builds connections to the R/2 host via LU6.2 using services of the SNA communication subsystem. Several communications requests can be processed via one SAP Gateway.

Prerequisites

With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - There must be a SAP Gateway in use that supports SNA.
  - There must be an SNA communication subsystem.
  - The called function must be an ABAP function module, which has the “remote” ID in the function library.

For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

- Using the CPI-C interface
  - There must be a SAP Gateway in use that supports SNA.
  - There must be an SNA communication subsystem.
  - The ABAP target program on the R/2 host must contain the form routine specified in the connect data.
  - The calling ABAP program must build the data for the logon to the target system, convert it to EBCDIC and receive the response (possibly a denial) from the target system.
  - The side info table TXCOM must be configured in the calling R/3 System. For this, use transaction code SM54.

For details on the configuration of TXCOM, refer to the relevant topic in the following documentation:

- Using the CPI-C interface
  - There must be a SAP Gateway in use that supports SNA.
  - There must be an SNA communication subsystem.
  - The ABAP target program on the R/2 host must contain the form routine specified in the connect data.
  - The calling ABAP program must build the data for the logon to the target system, convert it to EBCDIC and receive the response (possibly a denial) from the target system.
From R/3 to R/2

- The side info table TXCOM must be configured in the calling R/3 System. For this, use transaction code SM54.

For details on the configuration of TXCOM, refer to the relevant topic in the following documentation:

BC SAP Communication: Configuration [Extern]

**Process flow**

- Using the RFC interface:
  
  Once the above requirements have been met, select the target system with the Transaction SM59.

- Using the CPI-C interface
  
  For details on CPI-C calls, refer to the topic CPI-C Implementation in ABAP [Seite 51].
From R/2 to R/3

Purpose
An ABAP program in an R/2 System starts an ABAP program in an R/3 System and exchanges data with this program.
You can start the target program from the R/2 host via an SAP communications program.

Prerequisites
The SAP communications program and the SAP Gateway must be on a UNIX computer known in the SNA network. The target program, on the other hand, can be on a UNIX computer not in the SNA network.
Side info files must be configured on the UNIX computer known in the SNA network and on the Gateway host.
With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - The side info tables XCOM and RFCD must be configured.
  - The destination of the function call must match the corresponding entry in XCOM and RFCD.
  - The called function must exist as a function module in the ABAP function library and have the “remote” ID.

For more details on the RFC interface, refer to the documentation Remote Communications.

- Using the CPI-C interface
  - The SAP Gateway must be in use.
  - The communications program must be known to the SNA software.
  - The User ID and the work directory of the program started are manufacturer-specific:

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>The program runs under the user ID specified in the SNA definition, in the home directory of this user ID.</td>
</tr>
<tr>
<td>HP</td>
<td>The program runs under the user ID sna (ID, under which the SNA kernel runs), in the directory containing the program.</td>
</tr>
</tbody>
</table>

IMS:
If you log on to an IMS security system, the session must not be closed. For this reason, the environment variable SAP_KEEP_SESSION was introduced.
For SAP_KEEP_SESSION=1, the session is retained despite the statement COMMUNICATION DEALLOCATE.
Process flow

1. The calling ABAP program reads the TP name in the XCOM table and starts the SAP communications program *gwhost* (for CICS) or *gwims* (for IMS) under this name on a computer known in the SNA network.

2. The communications program sets up a *Conversation* with the SAP Gateway.

3. The SAP Gateway starts the actual target program. The computer, on which it is located, does not have to be known in the SNA network.

The communications program simply passes on the data during the CPI-C dialog.

For details on the configuration of side info files, refer to the following topic in the documentation:

- *Side Information in R/2 on the MVS/VSE Host: XCOM [Extern]*
- *Parameters on SNA Subsystem Platform with R/2 [Extern]*
Communication Between R/3 and R/2 (BS2000)

Purpose
The following topics provide an overview about the prerequisites for the partner systems. These prerequisites depend on whether the calling system is R/3 or R/2.

From R/2 to R/3 [Seite 28] [Seite 26]
From R/3 to R/2 [Seite 26] [Seite 28]

As of R/2 Release 5.0D, you can also use an RFC call instead of a CPI-C call.
From R/3 to R/2

Purpose
An ABAP program in an R/3 System uses an RFC or CPI-C call to start an ABAP program in an R/2 System under DCAM or UTM on a BS2000 host, and exchanges data with this program.

Prerequisites
- Using the RFC interface:
  - The SAP Gateway must run under DCAM in BS2000.
  - The called function must be an ABAP function module, which has the “remote” ID in the function library.
  For more details on the RFC interface, refer to the documentation Remote Communications [Extern].
- Using the CPI-C interface
  - The SAP Gateway must run under DCAM in BS2000.
  - The target ABAP program must contain the form routine specified in the connect data.
  - If you are using the CPI-C calls from ABAP, the calling ABAP program must build the data for the logon to the target system, convert it to EBCDIC and receive the response (possibly a denial) from the target system.
  - The side info table TXCOM must be configured in the calling R/3 System. For this, use transaction code SM54.
  For details on the configuration of TXCOM, refer to the relevant topic in the following documentation:
    BC - SAP Communication: Configuration [Extern]
  - If there is a side info file SAPGW.DATA.SIDEINFO on the BS2000 host, an application can only be reached if the side info file contains an entry for this application.
  For details on configuration, please refer to the documentation BC - SAP Communication: Configuration under the topic Connection Setup to the R/2 System [Extern]

Process flow
- Using the RFC interface:
  Once the above requirements have been met, select the target system with the Transaction SM59.
  For more details on the RFC interface, refer to the documentation Remote Communications [Extern].
- Using the CPI-C interface
  For details on configuration, please refer to the documentation BC - SAP Kommunikation: Configuration under the topic Connection Setup to the R/2 System
From R/2 to R/3

Purpose
An ABAP program in an R/2 System starts an ABAP program in an R/3 System and exchanges data with this program.

You can start the target program from the R/2 host via an SAP communications program.

Prerequisites
With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - The side info tables XCOM and RFCD must be configured.
    The destination of the function call must match the corresponding entry in XCOM and RFCD.
  - The called function must exist as a function module in the ABAP function library and have the “remote” ID.

  For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

- Using the CPI-C interface
  - The SAP Gateway must be in use.

  Special Features under BS2000
  The tasks SAPGWHO (program gwhost) are started by the R/2 System at initialization. These tasks build the connection to an SAP Gateway under UNIX (not under BS2000).

Process flow
In BS2000 the SAPGWHO jobs (program gwhost) perform the functions of the communications program when connection is set up by the R/2 System.

The process consists of the following steps:

1. The calling ABAP program reads the TP name in table XCOM and starts the SAPGWHO jobs.
2. The communications program sets up a Conversation with a SAP Gateway on the UNIX computer.
3. The SAP Gateway starts the actual target program.

  Side info files must be configured on the UNIX computer and on the BS2000 host.

For details on the configuration of side info files, refer to the documentation BC SAP Communication: Configuration under the following topic: Connection Setup by the R/2 System [Extern].
Communication Between R/3 and an Externally Registered Program

Purpose

The SAP program interfaces allow an R/3 System to communicate and exchange data with an external program.

As well as the external programs that are always started anew when they are needed, there is also the new program type Registered Program [Seite 39].

The following topics provide an overview about the prerequisites for the partner systems.

These prerequisites depend on whether R/3 is the calling system or the called system:

- From an External Program to R/3 [Seite 36] [Seite 31]
- From R/3 to an External Program (registered program) [Seite 31] [Seite 36]
From R/3 to an External Program (registered program)

Purpose
An ABAP program in an R/3 System starts a non-SAP program on another system and exchanges data with this program.
The external program may also be a Registered Program [Seite 39].
Communication is possible with the following external target programs:
- ANSI-C programs on UNIX platforms by RFC or CPI-C call
- Visual Basic programs on Windows and WindowsNT platforms by RFC

Prerequisites / Process flow
With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - Choose the target system using transaction SM59.
  - The called function must be an ABAP function module, which has the “remote” ID in the function library.
  - A C target program must have the following structure:
    : 
    
    #include "saprfc.h"
    :
    main(int argc,char **argv)
    {
    :
    Rfc_Handle handle;
    handle=RfcAccept(argv)
    :
    }
    To link the C program, use the RFC library librfc.a.
  - A Visual Basic target program must be structured as follows:
    :
    Sub Main()
    gCommand$ = Command$
    :
    hRfc = RfcAcceptExt(gCommand$)
    :
    End Sub
    To link the Visual Basic program, use the librfc.lib (Windows) or ntlibrfc.lib libraries.

The following DLLs are used:
- librfc.dll
- librfc2.dll
From R/3 to an External Program (registered program)

- nidll.dll (for Windows only)
  For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

- Using the CPI-C interface
  - Table TXCOM in the calling R/3 System must contain the following parameter values:
    Symbolic destination
    Logical unit of the partner: Host name
    Transaction program of the partner
    Communication types
    
    E  Partner is an external program
    R  Partner is a registered program
    
    DEST1 is0001 cpict2  E

    Use transaction code SM54 to maintain TXCOM.
    For details on the configuration of TXCOM, refer to the relevant topic in the following documentation:
    BC SAP Communication: Configuration [Extern]

- A C target program must have the following structure.

| Non-registered program | Registered program |
To link the C program, use the function library `pictlib.o`.

The function `SAP_CMACCP` [Seite 112] is used to pass the parameters needed to establish the connection to the CPI-C interface.

If an error occurs, `SAP_CMALLC` gives a return code not equal to `CM_OK`. After calling `SAP_CMACCP`, you can use CPI-C programming as usual.

⚠️

The pointer passed when `SAP_CMACCP` is called cannot point to data in the stack.

This is because the pointer is also used in the subsequent CPI-C functions.

- If the gateway host and the target host are the same:

  **UNIX:**
  
  The program must be in the search path of the of the User ID of the SAP Gateway and be startable with the User ID. (HP-UX: Gateway host and target host must always be the same.)

  **WindowsNT:**
From R/3 to an External Program (registered program)

The program must be in the work directory of the SAP Gateway.

**OS/2:**

The program is in the work directory of the SAP Gateway or it is defined in the configuration file config.sys via the PATH variable.

- If the gateway host and the target host are not the same:

  The SAP Gateway starts the C program on the target host. The following conditions must be met:

  **UNIX:**

  The file .rhosts must be available in the HOME directory of the User ID of the SAP Gateway.

  The User ID, under which the SAP Gateway is running, must be known on the target host.

  The target program or a link to it must be available in the HOME directory.

<table>
<thead>
<tr>
<th>CPI-C:</th>
<th>limited to 8 characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC:</td>
<td>no limitation, relative or explicit path</td>
</tr>
</tbody>
</table>

**OS/2:**

The IP address of the respective partner computer must exist in the hosts file on both computers.

The program is in a path defined in the configuration file config.sys via the PATH variable.

- The started program can also be a shell script or a command procedure, which calls the actual CPIC-C program. In this case, all parameters of the shell script must be passed to the C program.

Example:

Command file for OS/2:

```bash
@REM **************************************************
@REM * CPICPGM.CMD for OS/2
@REM **************************************************
@SETLOCAL

@REM set up environment
@SET CPIC_TRACE=1
:
@REM start Remote Partner Program
D:\CPIC\CPICPGM.EXE %1 %2 %3
:
@ENDLOCAL
```

C shell script for UNIX:

```bash
#!/bin/csh
.
<Shell commands>
```
<cpic-c program> $argv

From an External Program to R/3

Purpose
A non-SAP program uses an RFC or CPI-C call to start an ABAP program in an R/3 System, and exchanges data with the ABAP program.

Prerequisites / Process flow
With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - A calling C program (ANSI-C) must be structured as follows:
    ```c
    #include "saprfc.h"
    main(int argc,char **argv)
    {
        Rfc_Handle handle;
        handle=RfcOpen(RFC_OPTIONS *options);
    }
    ```
    To link the C program, use the RFC library librfc.a.
  - A calling Visual Basic program must be structured as follows:
    ```vbs
    Sub Main()
    gCommand$ = Command$
    hRfc = RfcOpenExt(gCommand$)
    End Sub
    ```
    To link the Visual Basic program, use the librfc.lib (Windows) or ntlibrfc.lib libraries.

  The following DLLs are used:
  - librfc.dll
  - librfc2.dll
  - nidll.dll (for Windows only)

  - The called function must be an ABAP function module, which has the “remote” ID in the function library.

  For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

- Using the CPI-C interface
  - The SAP Gateway must be in use.
  - The calling program (ANSI-C) must be structured as follows:
To link the C program, use the function library `cpictlib.o`.

- **Parameterization**

  **Via SAP_CMINIT [Seite 111]:**

  For communication with the gateway, you must use the function SAP_CMINIT to pass the following parameters to the CPI-C interface:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2:</td>
<td>Gateway host</td>
</tr>
<tr>
<td>P3:</td>
<td>Gateway service</td>
</tr>
<tr>
<td>P4:</td>
<td>Protocol type/topology of the target program: C, I or E</td>
</tr>
</tbody>
</table>

  - C: Partner is the R/2 program
  - I: Partner is the R/3 program and can be reached via TCP/IP
  - E: Partner is an external program and can be reached via TCP/IP

  The necessary constants are defined in the C header `cpic.h`.

  If an error occurs, SAP_CMINIT delivers a return code not equal to CM_OK.

  **Side Info File:**

  You can also maintain these parameters in the side information file. In this case, call the function SAP_CMINIT as follows:

  ```c
  SAP_CMINIT(argv, (PCPIC_CHAR)0, (PCPIC_CHAR)0, NO_PROTOCOL);
  ```

  After calling SAP_CMINIT you can use CPI-C programming as usual.

  You can also specify some parameters with SAP_CMINIT and read some from the side information file.

  - You must include target system and platform-dependent entries in the side information file.

  For details on the configuration of the side information file, refer to the relevant topic in the following documentation:

  **BC_SAP Communication: Configuration [Extern]**
If timeouts occur when establishing a connection, you can use the environment variable CPIC_TIMEOUT to extend the wait time. CPIC_TIMEOUT specifies the time in seconds, which the gateway waits for the external program logon.

⚠️

The pointer passed when SAP_CMACCP is called cannot point to data in the stack. This is because the pointer is also used in the subsequent CPI-C functions.
Registered Program

Definition
A registered program is an external program that logs on to the Gateway once (registration); the Gateway then starts a process and sets up a TCP/IP line. The program is then executed, if required.

Use
Using a registered program is particularly advantageous for performance if it is a program that is used very frequently. It avoids having to repeatedly start a new process.

Integration
A registered program can be stopped by another program with the CANCEL call (i.e. registration is cancelled), if it is not active at that moment.
Communication Between R/2 Systems

Purpose
Between R/2 Systems, program-to-program communication is possible with the following restrictions:

Restrictions for MVS/VSE hosts
Communication is only possible on MSE/VSE hosts when CICS is used as the data communications system. Local communication in R/2 is not possible because CICS does not support a local conversation via SNA-LU6.2.

Restrictions for BS2000 hosts
Communication between R/2 Systems on BS2000 is possible if both systems are operated under UTM.

Prerequisites
With this constellation, you must note the following guidelines:

- When using the RFC interface (from 5.0D):
  - The side info tables XCOM and RFCD must be configured.
    - The destination of the function call must match the corresponding entry in XCOM and RFCD.
  - The called function must be an ABAP function module, which has the "remote" ID in the function library.

- Using the CPI-C interface
  - The calling ABAP program must build the data for the logon to the target system, convert it to EBCDIC and receive the response (possibly a denial) from the target system.
  - The table XCOM must be configured in the calling R/2 System.
    For details on the configuration of XCOM, refer to the relevant topic in the following documentation:
    - BC  SAP Communication: Configuration [Extern]
    - The target ABAP program must contain the form routine specified in the connect data.

Process flow
For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

For details on CPI-C calls, refer to the topic CPI-C-Implementation in ABAP [Seite 51].
Communication Between R/2 and an Externally Registered Program

Purpose

Communication between the R/2 System and external programs is also supported by the SAP program interfaces.

The following topics provide an overview about the prerequisites for the R/2 host and the partner computer.

These prerequisites depend on whether R/2 is the calling system or the called system:

From R/2 to an External Program (registered program) [Seite 42]

From an External Program to R/2 [Seite 46]

As of R/2 Release 5.0D, you can also use an RFC call instead of a CPI-C call.

The following limitation applies to an MVS/VSE host:

- CICS only as the DC system (at present)
- IMS as of Version 4.1 for complete LU6.2 support
From R/2 to an External Program (registered program)

Purpose
An ABAP program in an R/2 System starts an ABAP program in an R/3 System or an non-SAP program, and exchanges data with this program.
You can start the target program from the R/2 host via an SAP communications program.
Communication is possible with the following external target programs:

- ANSI-C programs on UNIX platforms by RFC or CPI-C call
- Visual Basic programs on Windows and WindowsNT platforms by RFC

Prerequisites
With this constellation, you must note the following guidelines:

- Using the RFC interface:
  - The side info tables XCOM and RFCD must be configured.
    The destination of the function call must match the corresponding entry in XCOM and RFCD.
  - The called function must exist as a function module in the ABAP function library and have the "remote" ID.
  - A C target program must have the following structure:
    ```c
    #include "saprfc.h"
    main(int argc,char **argv)
    {
      Rfc_Handle handle;
      handle=RfcAccept(argv);
    }
    ```
    To link the C program, use the RFC library `librfc.a`.

- A Visual Basic target program must be structured as follows:
  ```vb
  Sub Main()
    gCommand$ = Command$
    hRfc = RfcAcceptExt(gCommand$)
  End Sub
  ```
  To link the Visual Basic program, use the `librfc.lib` (Windows) or `ntlibrfc.lib` libraries.

The following DLLs are used:
- `librfc.dll`
• librfc2.dll
• nidll.dll (for Windows only)

For more details on the RFC interface, refer to the documentation Remote Communications [Extern].

• Using the CPI-C interface
  – The SAP Gateway must be in use.
  – The target program must have exactly the same structure as described in the previous section.
  – A C target program must have the following structure:

<table>
<thead>
<tr>
<th>Non-registered program</th>
<th>Registered Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; poss. host type define, see Define Variables for Host Types [Seite 130] &gt;</td>
<td>&lt;poss. host type define &gt;</td>
</tr>
<tr>
<td>define SOCK 1</td>
<td>define SOCK 1</td>
</tr>
<tr>
<td>include &quot;cpic.h&quot;</td>
<td>include &quot;cpic.h&quot;</td>
</tr>
<tr>
<td>main(int argc,char **argv)</td>
<td>main(int argc,char **argv)</td>
</tr>
<tr>
<td>{</td>
<td>}</td>
</tr>
<tr>
<td>SAP_CMACCP(argv);</td>
<td>SAP_CMREGTP(argv);</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CMAACC(..);</td>
<td>CMAACC(..);</td>
</tr>
</tbody>
</table>

To link the C program, use the function library cpictlib.o.

The function SAP_CMACCP [Seite 112] is used to pass the parameters needed to establish the connection to the CPI-C interface.
From R/2 to an External Program (registered program)

If an error occurs, SAP_CMACCP [Seite 112] gives a return code not equal to CM_OK. After calling SAP_CMACCP, you can use CPI-C programming as usual.

⚠️
The pointer passed when SAP_CMACCP is called cannot point to data in the stack.
This is because the pointer is also used in the subsequent CPI-C functions.

- The communications program must be known to the SNA software.
- The User ID and the work directory of the program started are manufacturer-specific:

| IBM: | The program runs under the user ID specified in the SNA definition, in the home directory of this user ID. |
| HP:  | The program runs under the user ID sna (ID, under which the SNA kernel runs), in the directory containing the program. |

💡 Special Features under BS2000

The tasks SAPGWHO (program gwhost) are started by the R/2 System at initialization. These tasks build the connection to an SAP Gateway under UNIX (not under BS2000).

⚠️ IMS:

If you log on to an IMS security system, the session must not be closed. For this reason, the environment variable SAP_KEEP_SESSION was introduced.

For SAP_KEEP_SESSION=1, the session is retained despite the statement COMMUNICATION DEALLOCATE.

Process flow

The communication process depends on the host you are using:

- R/2 System on MVS/VSE Host
- R/2 System on BS2000 Host

R/2 System on MVS/VSE Host

The SAP communications program and the SAP Gateway must be on a UNIX host known in the SNA network. The target program, on the other hand, can be on a UNIX host not in the SNA network.

The process consists of the following steps:

1. The calling ABAP program reads the TP name in the XCOM table and starts the SAP communications program gwhost (for CICS) or gwims (for IMS) under this name on a host known in the SNA network.

2. The communications program sets up a Conversation with the SAP Gateway.

3. The SAP Gateway starts the actual target program. The host, on which it is located, does not have to be known in the SNA network.
The communications program simply passes on the data during the CPI-C dialog.

Side info files must be configured on the UNIX host known in the SNA network and on the Gateway host.

For details on the configuration of side info files, refer to the following topic in the documentation:

---

* Side Information in R/2 on the MVS/VSE Host: XCOM [Extern]
* Parameters on SNA Subsystem Platform with R/2 [Extern]

---

**R/2 System on BS2000 Host**

In BS2000 the SAPGWHO jobs (program `gwhost`) perform the functions of the communications program when connection is set up by the R/2 System.

The process consists of the following steps:

1. The calling ABAP program reads the TP name in table XCOM and starts the SAPGWHO jobs.
2. The communications program sets up a *Conversation* with a SAP Gateway on the UNIX host.
3. The SAP Gateway starts the actual target program.

   Side info files must be configured on the UNIX host and on the BS2000 host.

For details on the configuration of side info files, refer to the following topic in the documentation:

---

* Connection Setup by the R/2 System:
From an External Program to R/2

Purpose
A non-SAP program uses an RFC or CPI-C call `CMINIT/CMALLC` [Seite 89] to start an ABAP program in an R/3 System, and exchanges data with the ABAP program.

Prerequisites / Process flow
With this constellation, you must note the following guidelines:

- Using the RFC interface:
  The RFC interface is available for R/2 as of Release 5.0D.
  - A calling C program (ANSI-C) must be structured as follows:
    ```
    #include "saprfc.h"
    
    main(int argc, char **argv)
    {
    
    Rfc_Handle handle;
    handle=RfcOpen(RFC_OPTIONS *options)
    }
    ```
  To link the C program, use the RFC library `librfc.a`.
  - A calling Visual Basic program must be structured as follows:
    ```
    Sub Main()
    gCommand$ = Command$
    
    hRfc = RfcOpenExt(gCommand$)
    
    End Sub
    ```
  To link the Visual Basic program, use the `librfc.lib` (Windows) or `ntlibrfc.lib` libraries.
  The following DLLs are used:
  - `librfc.dll`
  - `librfc2.dll`
  - `nidll.dll` (for Windows only)
  - The called function must be an ABAP function module, which has the “remote” ID in the function library.

For more details on the RFC interface, refer to the documentation `Remote Communications [Extern]`.

- Using the CPI-C interface
  - The SAP Gateway must be in use.
The calling program (ANSI-C) must be structured as follows:

```
#define SOCK 1
#include "cpic.h"

main(int argc,char **argv)
{
    SAP_CMINIT(argv,<P2>,<P3>,<P4>);
    CMINIT(..);
}
```

To link the C program, use the function library `cpictlib.o`.

Parameterization

Via **SAP_CMINIT [Seite 111]**:

For communication with the gateway, you must use the function `SAP_CMINIT` to pass the following parameters to the CPI-C interface:

- **P2**: Gateway host
- **P3**: Gateway service
- **P4**: Protocol type/topology of the target program: C, I or E
  - C: Partner is the R/2 program
  - I: Partner is the R/3 program and can be reached via TCP/IP
  - E: Partner is an external program and can be reached via TCP/IP

The necessary constants are defined in the C header `cpic.h`.

If an error occurs, `SAP_CMINIT` delivers a return code not equal to `CM_OK`.

Side Info File:

You can also maintain these parameters in the side information file. In this case, call the function `SAP_CMINIT` as follows:

```
SAP_CMINIT(argv,(PCPIC_CHAR)0,(PCPIC_CHAR)0,NO_PROTOCOL);
```

After calling `SAP_CMINIT` you can use CPI-C programming as usual.

You can also specify some parameters with `SAP_CMINIT` and read some from the side information file.

You must include target system and platform-dependent entries in the side information file.

For details on the configuration of the side information file, refer to the relevant topic in the following documentation:

**BC SAP Communication: Configuration [Extern]**
From an External Program to R/2

If timeouts occur when establishing a connection, you can use the environment variable CPIC_TIMEOUT to extend the wait time. CPIC_TIMEOUT specifies the time in seconds, which the gateway waits for the external program logon.

⚠️

The pointer passed when SAP_CMACCP is called cannot point to data in the stack. This is because the pointer is also used in the subsequent CPI-C functions.
Communication Between C Programs

Purpose

A C program uses a CPI-C call **CMINIT/CMALLC** [Seite 89] to start a C program on another host, and exchanges data with this program via CPI-C.

RFC is not supported for this constellation.

Prerequisites / Process flow

With this constellation, the following requirements must be met:

- The SAP Gateway must be in use.
- If no local side information file was maintained, you must make the following entries in the side information file on the gateway host:
  
  DEST=<symbolic destination>
  LU=<name of the target host>
  TP=<name of the target host>
- The target program (ANSI-C) must be structured as follows:

  ```
  < poss. host type define, see Define Variables for Host Types [Seite 130] >
  
  #define SOCK 1
  #include "cpic.h"
  
  main(int argc,char **argv)
  {
    
    CMACCP(..);
    
  }
  
  ```

  To link the C program, use the function library `cpictlib.o`.
- The calling program (ANSI-C) must be structured as follows:

  ```
  < poss. host type define, see Define Variables for Host Types [Seite 130] >
  
  #define SOCK 1
  #include "cpic.h"
  
  main(int argc,char **argv)
  {
    
    CMINIT(..);
    
  }
  ```
Communication Between C Programs

To link the C program, use the function library `cpictlib.o`. 
CPI-C Implementation in ABAP

Purpose

Communication statements of the *CPI-C Starter Set* are implemented in the ABAP programming language. They are suitable for setting up simple communication.

This allows an ABAP program to actively set up a connection to one or more external application programs in order to send data directly to these programs.

Conversely, an external program can set up a connection to an SAP System and send a start request for an ABAP program. A correct logon to the SAP System is necessary for this type of connection setup.

The ABAP statements are initiated by the key word COMMUNICATION. The key word is followed by a function statement, which determines the communication operation to be performed. The function statement is followed by further parameters.

Features

CPI-C calls and their meaning

<table>
<thead>
<tr>
<th>CPI-C Call in C</th>
<th>CPI-C Call in ABAP</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMINIT</td>
<td>COMMUNICATION INIT</td>
<td>Initialize connection</td>
</tr>
<tr>
<td>CMALLC</td>
<td>COMMUNICATION ALLOCATE</td>
<td>Set up connection</td>
</tr>
<tr>
<td>CMACCP</td>
<td>COMMUNICATION ACCEPT</td>
<td>Accept connection</td>
</tr>
<tr>
<td>CMSEND</td>
<td>COMMUNICATION SEND</td>
<td>Send Data</td>
</tr>
<tr>
<td>CMRCV</td>
<td>COMMUNICATION RECEIVE</td>
<td>Receive data</td>
</tr>
<tr>
<td>CMDEAL</td>
<td>COMMUNICATION DEALLOCATE</td>
<td>Close the connection</td>
</tr>
</tbody>
</table>

- The return code of the statement in the field SY-SUBRC is made available for all variants.
  
  As of R/2 Release 5.0 and in R/3, the parameter RETURNCODE <rc> is supported as an option. With this, you receive a return value.

- Symbolic values (constants, return codes) are defined in the INCLUDE member RSCPICDF (hexadecimal).

- Only the most important return values (return codes) are listed in the description of the statements.

  If you want to perform a detailed problem analysis, you should also analyze the entries in the system log and the system-specific traces.

See also:

- Agreements Between Sender and Recipient [Seite 53]
- CPI-C Implementation in ABAP [Seite 67]
CPI-C Implementation in ABAP
Agreements Between Sender and Recipient

The information exchange process is controlled by both communications partners. All the necessary agreements concerning type, method and the contents of the information flow must be agreed:

- **Send/Receive Mode [Seite 54]**
- **Synchronization [Seite 55]**
- **Size of Transfer Units [Seite 56]**
- **Data Types and Structures [Seite 57]**
- **Data Conversion [Seite 58]**
- **Remotely Attachable ABAP Program [Seite 59]**
- **SAP Logon Log for External CPI-C Programs [Seite 62]**
- **Establishing a Connection via an ABAP Program [Seite 65]**
Setting the Send/Receive Mode

Purpose

CPI-C works in two-way-alternate mode. This means that at any one time, only one of the communications partners is authorized to send data. Both communications partners must therefore agree on how to alternate between send and receive (possibly depending on the DC system).

Process flow

The program, which builds the conversation [Extern], has send authorization at first. It can call the following functions:

- COMMUNICATION SEND
  Send Data
- COMMUNICATION RECEIVE
  Return the send authorization and wait for data from the partner
- COMMUNICATION DEALLOCATE
  Close the connection

By calling the receive command, you can pass the send authorization to the partner program, for example, to request an acknowledgement. The receiving program should therefore always check whether it has received the send authorization (see parameter STATUSINFO under COMMUNICATION RECEIVE: Receive Data [Seite 76]).
Synchronization

Purpose

The ABAP communications interface (based on the CPI-C Starter Set) has no explicit functions for synchronization of both partners. The Advanced Function Calls CMCFMI/CMCFMID are not implemented in ABAP.

Both communications partners must perform the synchronization with send and receive commands on the application level.

Synchronization is necessary because a send request does not necessarily trigger the immediate sending of data.

Process flow

Different communications subsystems buffer data first and only send it physically when certain buffer limits are exceeded, a status change is made or the connection is closed.

Synchronization means:

- Sending as yet unsent data to the partner (flush)
- Requesting an acknowledgement from the partner
- The partner sending an acknowledgement
- Waiting for the acknowledgement to be received

The easiest method involves passing the send authorization to the partner program after sending one or more data blocks in order to request an acknowledgement. It is recommendable to use a RECEIVE call after no more than 7 SEND calls. The larger the data blocks, the less SEND calls you should use before the next RECEIVE call. The partner can return the send authorization (acknowledgement) immediately. In addition, a response (data) can be sent in order to differentiate between positive and negative acknowledgements.

Certain restrictions apply for setting up communication via LU6.2 from IMS (up to Version 3.1).

When changing from COMMUNICATION SEND to COMMUNICATION RECEIVE, any unsent data is transferred to the communications partner.

For each COMMUNICATION SEND there must be at least one COMMUNICATION RECEIVE in the partner system.
Setting the Size of Transfer Units

You must define the size of the send/receive buffer within the application program in accordance with the size of the data to be transmitted. If you have agreed to use different buffer sizes, this may result in data loss if the receiver buffer is too small. The CPI-C implementation in ABAP in R/2 does not allow remaining data to be collected with receive commands. R/3 does not have this restriction.

To achieve optimum performance, the transfer unit size you choose should be as large as possible (parameter LENGTH under COMMUNICATION SEND: Send Data [Seite 74]).

After you have received data, you should check whether the data in the buffer is complete (see parameter DATAINFO under COMMUNICATION RECEIVE: Receive Data [Seite 76]).
Selecting Data Types and Structures

The receive program must be able to recognize which type of data it can receive and whether this data is complete and correct.

An additional logon log can also be agreed between the sender and the receiver, for example, in order to request data again, to provide notification of a processing problem, or simply to send an acknowledgement to the partner. This type of control sequence must be distinguishable from pure data, for example, by defining a special log header (type, record type, number...).

When organizing communication between heterogeneous systems, please note that representation of one data type can be different in each system, and the receiver of this format may not be able to process the format directly. Examples of this are whole numbers, packed numbers and floating point numbers.

In this case, you should consider defining a common transfer format. The simplest form of this would be conversion of data into a character string.
Data Conversion

Between communications systems, which work with the ASCII character set and systems using the EBCDIC character set, data conversion must normally be performed by the application program, unless the partner system is used only for archiving binary data.

It may be necessary to make special considerations for country-specific code pages. You can build your own conversion tables within an ABAP program and access them using the TRANSLATE command.

```
TRANSLATE <buffer> USING <tab>.
TRANSLATE <buffer> FROM CODEPAGE <code_1>
     TO CODEPAGE <code_2>.
```
Remotely Attachable ABAP Program

Prerequisites

To process an external communications request, it is necessary to establish a connection to a service transaction of the SAP Basis system.

In the R/2 System, this connection is established to transaction X1SA (BS2000, CICS) or xxxX1SA (IMS).

In the R/3 System, the connection is established to the SAP dispatcher via the SAP Gateway.

The Service transaction needs a logon sequence to identify the client, the user and the ABAP program. The SAP Basis system then starts the ABAP program. The activated ABAP program must be executable (type 1 or M) and have at least one FORM routine.

This logon is part of the SAP logon protocol CPI-C. If logon is performed incorrectly, the SAP Basis system closes the connection with an appropriate error message.

Procedure

A subroutine (FORM routine), which is specified in the program start request, must be defined in an ABAP program. This routine is triggered automatically. Within the FORM routine, PERFORM can be used to call further subroutines (including external ones) or to establish communications links with other partners.

The main part of an ABAP program is not executed for CPI-C connections. It can therefore contain a different processing logic for normal online operation.

An activated FORM routine can only perform one ACCEPT command, after which it is in receive status.
Remote Start Request

An ABAP program can contain several such subroutines, which can be accessed via different logon sequences. This allows you to build up a function library for CPI-C processing.

**Special Features of ABAP in R/2**

If you want to create a remotely attachable ABAP CPI-C program in an R/2 System, you must know that some ABAP key words are not permitted or are processed differently than in normal online operation. This is because no online terminal is available for screen output, as for an update program. You must note the following:

**Screen change**

A session or screen change cannot be initiated within an SAP system during a conversation [Extern]. This means that you cannot call subsequent key words in the remote ABAP program, because this causes the conversation and the program to terminate.

- CALL MENU...
- CALL SCREEN...
- CALL DIALOG...
- CALL TRANSACTION... (without USING...) (zulässig with Zusatz "USING <Dynprotabelle> Mode 'S'/'N' ", Zusatz verfügbar ab R/2 5.0)
- SUBMIT REPORT... (permitted from R/2 Release 5.0)
TRANSFER
You can use the TRANSFER command to write data to the SAP spool component (from R/2 Release 5.0)

WRITE
The key word WRITE is ignored or redirected to a spool member if the statement NEW-PAGE PRINT ON NO DIALOG is specified first (from R/2 Release 4.3J, 4.4D, 5.0).

BREAK-POINT
The statement BREAK-POINT writes log informatin to the SAP system log (from R/2 Release 4.3J, 4.4D, 5.0).

MESSAGE
Messages of type S, I or W are ignored. Messages of type E or A cause a program termination and an entry in the system log.
SAP Logon Protocol for External CPI-C Programs

Definition
The information exchanged between an external program and the transaction program (R/2: X1SA, R/3: Dispatcher service) for the logon is determined in the SAP system.

The SAP logon protocol consists of the protocol header (12 bytes) and the actual logon to the SAP System.

Structure

General Protocol Header
The protocol header, the first 12 bytes of the logon sequence, has the following structure:

Header of The SAP Logon Protocol

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>9</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPIC</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Protocol header description:

<table>
<thead>
<tr>
<th>External logon description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestID</td>
<td>Request ID for internal SAP communications administration&lt;br&gt;CONN = Logon/start request&lt;br&gt;APPC = Positive acknowledgement&lt;br&gt;FREE = negative acknowledgement/termination message</td>
</tr>
<tr>
<td>Type</td>
<td>Logon protocol type&lt;br&gt;CPIC = for ABAP CPI-C communication</td>
</tr>
<tr>
<td>ModeNo</td>
<td>Mode number within the SAP system&lt;br&gt;1 = Default value</td>
</tr>
<tr>
<td>Reserved</td>
<td>’ ’ = Default value</td>
</tr>
</tbody>
</table>
External Logon to the SAP System

For a correct logon, a CONNECT message must be sent to the transaction program in the desired SAP target system. This logon/start request must always be sent in EBCDIC code.

Logging on to The SAP System

Example:

```
CONN CPIC 1
001 USER MYPASS D S MYABAP00 MYFORMABC
```

External logon to the SAP system

<table>
<thead>
<tr>
<th>External logon description:</th>
<th>Description</th>
<th>No. of characters</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol header</td>
<td></td>
<td>12</td>
<td>CONNCPIC1</td>
</tr>
<tr>
<td>Client</td>
<td>Client in the SAP System</td>
<td>3</td>
<td>001</td>
</tr>
<tr>
<td>User name</td>
<td></td>
<td>12</td>
<td>USER</td>
</tr>
<tr>
<td>Password</td>
<td>User password (PASS not permitted)</td>
<td>8</td>
<td>MYPASS</td>
</tr>
<tr>
<td>Language</td>
<td>For messages from the SAP system</td>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>D = German</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E = English</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Not used for CPI-C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ABAP program</td>
<td>Partner program</td>
<td>08</td>
<td>MYABAP00</td>
</tr>
<tr>
<td>ABAP form</td>
<td>Subroutine</td>
<td>30</td>
<td>MYFORMABC</td>
</tr>
</tbody>
</table>
SAP Logon Protocol for External CPI-C Programs

- If you are using an external security system (RACF, ACF2, TOP SECRET..), transfer of user and client ID via SAP SignOn Exit (SONEX) for CPI-C connections is not supported. A valid user/password combination must therefore be available in the SAP system.
- From R/2 Release 5.0 and in R/3, only specific user IDs are permitted (user master record type CPIC).

Messages From the SAP System

After a CONNECT request is sent from an external system, the SAP system sends a response. The type of message depends on whether the logon was correct or not.

Positive response

After a connection request (CONN...) is sent with valid logon data, the SAP system returns the request to start CPI-C data transfer. This acknowledgement is always sent in EBCDIC format.

Within the SAP system, the specified ABAP FORM routine has already been activated and performed up to the first RECEIVE statement. This routine then waits for data from the partner.

Positive response: APPCCPIC1

Negative response

After a connection request (CONN...) is sent with invalid logon data, the SAP system terminates the conversation and sends an error message if the request had a valid format.

The negative response from the SAP system consists of the general protocol header beginning with FREE..., an error number and an error text (see the second example in the above illustration).

Negative response:

<table>
<thead>
<tr>
<th>External logon description</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol header</td>
<td>(12)</td>
<td>FREE</td>
</tr>
<tr>
<td>Error number</td>
<td>SAP error number from table 100 (5)</td>
<td>38110</td>
</tr>
<tr>
<td>Error message</td>
<td>SAP error text from table 100 (5)</td>
<td>ABAP program... not found</td>
</tr>
</tbody>
</table>
Establishing a Connection via an ABAP Program

Purpose
You can activate one or more connections from an ABAP program, and start an external (transaction) program, in the same way as you start an ABAP program from an external system. Certain restrictions apply for setting up communication via LU6.2 from IMS (up to Version 3.1).

Process flow
An external partner program is accessed in accordance with the CPI-C standard as a symbolic name (symbolic destination). This name is used during the initialization call (INIT) to determine the communications parameters, which are needed to establish the connection (ALLOCATE). You must maintain these parameters in the side information table (R/2 host: XCOM, R/3: TXCOM). Amongst other things, they contain system-specific information.

Once the connection is established, the program has send status. Data can now be sent to the partner.

Parallel Connections
If you want to establish several parallel connections, you must ensure that the following call sequence is adhered to:

COMMUNICATION INIT
COMMUNICATION ALLOCATE
...
COMMUNICATION INIT
COMMUNICATION ALLOCATE
...

The sequence COMMUNICATION INIT... COMMUNICATION INIT... COMMUNICATION ALLOCATE... COMMUNICATION ALLOCATE is not permitted because it causes the first defined communications parameters to be overwritten.

Restricted Use of Key Words (R/2)
The ABAP key words listed below cannot be called during a communications session because this causes the conversation to be terminated. The reason for this lies in the internal SAP administration for mode and screen changes.

- CALL MENU
- CALL DIALOG
- CALL SCREEN
- CALL TRANSACTION... (without USING...)
- SUBMIT REPORT
- LEAVE
- MESSAGE
Establishing a Connection via an ABAP Program

Debugging (R/2)

When a BREAKPOINT is reached, field contents can be displayed, but the connection is also broken so that a screen change can take place.

Debugging is not permitted under UTM because it violates the UTM rules for distributed transaction processing and causes the task to be terminated with 83Z/KS01.
CPI-C Implementation in ABAP

Purpose
The following statements are available for implementing communication at the CPI-C level in ABAP:

COMMUNICATION INIT: Initialization [Seite 68]
COMMUNICATION ALLOCATE: Set Up Connection [Seite 70]
COMMUNICATION ACCEPT: Accept Connection [Seite 72]
COMMUNICATION SEND: Send Data [Seite 74]
COMMUNICATION RECEIVE: Receive Data [Seite 76]
COMMUNICATION DEALLOCATE: Close Connection [Seite 78]

They are explained here individually.
The Return Codes [Seite 79] in ABAP are then listed.
COMMUNICATION INIT: Initialization

Use

The statement COMMUNICATION INIT initializes a connection. You must specify a symbolic name to identify the partner.

Integration

The statement COMMUNICATION INIT must always be followed by the statement COMMUNICATION ALLOCATE.

Activities

Syntax

COMMUNICATION INIT ID <conv_id>
DESTINATION <dest>
[ RETURNCODE <rc> ]

Parameter Values

<conv_id>
Conversation ID (output, type C(8))
An ID is returned to identify the conversation. You must specify the this ID in all subsequent communications statements for this conversation.

<dest>
Conversation ID (output, type C(8))
This name must also be in the side information file.

<rc>
Return code (output, type X(2))
Alternatively, you can check the value directly via SY-SUBRC.

Return Codes

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_OK</td>
<td></td>
</tr>
<tr>
<td>CM_PROGRAM_PARAMETER_CHECK</td>
<td>Entry in side info table incorrect</td>
</tr>
<tr>
<td>CM_PRODUCT_SPECIFIC_ERROR</td>
<td>• LU is not defined</td>
</tr>
<tr>
<td></td>
<td>• SAP transaction is in long-running status</td>
</tr>
<tr>
<td></td>
<td>• (UTM, additional system log message 818)</td>
</tr>
<tr>
<td></td>
<td>• No further ID available</td>
</tr>
</tbody>
</table>

DATA: CONV_ID(8) TYPE C,
COMMUNICATION INIT: Initialization

DEST(8) TYPE C VALUE 'PARTNER',
RC LIKE SY-SUBRC.

:  
COMMUNICATION INIT ID CONV_ID
DESTINATION DEST
RETURNCODE RC.
IF RC <> CM_OK....
COMMUNICATION ALLOCATE: Set up connection

Use

COMMUNICATION ALLOCATE builds a conversation [Extern] with the partner defined in the statement COMMUNICATION INIT. A session to the partner system is built and a start request for the partner program (for example, X1SA) is sent.

Activities

Syntax

COMMUNICATION ALLOCATE ID <conv_id>  
[ RETURNCODE <rc> ]

Parameter Values

<conv_id>
Conversation ID (input, type C(8))

<rc>
Return code (output, type X(2))
Alternatively, you can check the value directly via SY-SUBRC.

Return Codes

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_OK</td>
<td></td>
</tr>
<tr>
<td>CM_ALLOCATE_FAILURE_NO_RETRY</td>
<td>Resource or configuration problem (UTM: Also system log message 781)</td>
</tr>
<tr>
<td>CM_ALLOCATE_FAILURE_RETRY</td>
<td>Temporary problem (for example,: partner is not active)</td>
</tr>
<tr>
<td>CM_PROGRAM_PARAMETER_CHECK</td>
<td>Invalid conversation ID</td>
</tr>
<tr>
<td>CM_PROGRAM_STATE_CHECK</td>
<td>COMMUNICATION INIT call was not made</td>
</tr>
</tbody>
</table>

DATA: CONV_ID(8) TYPE C,  
      DEST(8) TYPE C VALUE 'PARTNER',  
      RC LIKE SY-SUBRC.  
COMMUNICATION INIT ID CONV_ID  
      DESTINATION DEST  
      RETURNCODE RC.  
:  
COMMUNICATION ALLOCATE ID CONV_ID
COMMUNICATION ALLOCATE: Set up connection

RETURNCODE RC.

IF RC <> CM_OK....
COMMUNICATION ACCEPT: Accept connection

**Use**

COMMUNICATION ACCEPT accepts the requested connection. Initializations are performed internally and an ID for the conversation is returned.

**Integration**

COMMUNICATION ACCEPT is only permitted within an ABAP FORM routine and can only be called once. After the function call, the program is in receive status.

**Activities**

**Syntax**

```
COMMUNICATION ACCEPT ID <conv_id> [ RETURNCODE <rc> ]
```

**Parameter Values**

- `<conv_id>`
  - *Conversation [Extern]ID (output, type C(8))*
  - To identify the conversation, an ID is returned. This ID must be specified in all subsequent communications statements for this conversation.

- `<rc>`
  - Return code (output, type X(2))
  - Alternatively, you can check the value directly via SY-SUBRC.

**Return Codes**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_OK</td>
<td></td>
</tr>
<tr>
<td>CM_PROGRAM_STATE_CHECK</td>
<td>• No connection request from a partner.</td>
</tr>
<tr>
<td></td>
<td>• Statement is not coded in a FORM routine.</td>
</tr>
</tbody>
</table>

```
DATA: CONV_ID(8) TYPE C,
     DEST(8) TYPE C VALUE 'PARTNER',
     RC LIKE SY-SUBRC.
FORM TEST
  COMMUNICATION ACCEPT ID CONV_ID
     RETURNCODE RC.
  IF RC <> CM_OK....
     : 
```
COMMUNICATION ACCEPT: Accept connection

ENDFORM.
COMMUNICATION SEND: Send Data

Use
Data is sent to the communications partner. The type and structure of the data can be agreed upon with the communications partner. The length of the CPI-C buffer is restricted to 32000 Bytes (maximum).

Due to this restriction, it is recommendable to work with a buffer of 28000 Bytes.

Integration
The program, which builds the conversation [Extern], has send authorization at first. If a program has send authorization, it can call the following functions:

- COMMUNICATION SEND
  Send Data
- COMMUNICATION RECEIVE
  Return the send authorization and wait for data from the partner
- COMMUNICATION DEALLOCATE
  Close the connection

Prerequisites
For CPI-C communication in an R/2 IMS environment via the LU6.1 Adapter, the length of the send buffer should not be greater than the LONG MESSAGE in IMS. (Recommendation: Send buffer length <= LONG MESSAGE - 100)

To achieve optimum performance, the transfer unit you choose should be as large as possible (see LENGTH parameter).

Activities
Syntax
COMMUNICATION SEND ID <conv_id>
  BUFFER <buf>
  [ LENGTH <slen>     ]
  [ RETURNCODE <rc>  ]

Parameter Values
<conv_id>
Conversation ID (input, type C(8))

<buf>
Data buffer (input, type C(?)) or structure: Range of the data to be sent

<slen>
COMMUNICATION SEND: Send Data

No. of characters to be sent (entry, type P)
If the length is not specified explicitly, the defined buffer length is used internally and filled with X'00' or blanks depending on the data type.

<rc>
Return code (output, type X(2))
Alternatively, you can check the value directly via SY-SUBRC.

Return Codes

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_OK</td>
<td></td>
</tr>
<tr>
<td>CM_PROGRAM_STATE_CHECK</td>
<td>Program is not in send status</td>
</tr>
<tr>
<td>CM_DEALLOCATED_ABEND</td>
<td>Partner has closed/interrupted the connection (UTM: Also SYSLOG message 781)</td>
</tr>
</tbody>
</table>

DATA: CONV_ID(8),
       BUF(72),
       SLEN TYPE P,
       RC LIKE SY-SUBRC.

BUF = 'Hello World'.
LEN = 11.
COMMUNICATION SEND ID CONV_ID
       BUFFER BUF
       LENGTH SLEN
       RETURNCODE RC.

IF RC <> CM_OK....
COMMUNICATION RECEIVE: Receive data

Use

The statement COMMUNICATION RECEIVE causes the program to wait for data from the partner. Incoming data is placed in the available buffer. In addition to the data, information on the completeness of the data and the communication status is reported.

If a program has send status and transmits COMMUNICATION RECEIVE, it passes the send authorization to the partner. The communications direction can be changed several times.

Activities

Syntax

COMMUNICATION RECEIVE ID <conv_id>
    BUFFER <buf>
    DATAINFO <di>
    STATUSINFO <si>
    [ HOLD ]
    [ RECEIVED <n> ]
    [ LENGTH <rlen> ]
    [ RETURNCODE <rc> ]

Parameter Values

<conv_id>
Conversation ID (input, C(8))

<buf>
Data buffer (output, type C(?) or structure): Data range/structure, which contains the received data

<di>
Data information (output, type X(4))
Data fully buffered, or not fully buffered if the buffer defined is too small.

CM_NO_DATA_RECEIVED
CM_COMPLETE_DATA_RECEIVED
CM_INCOMPLETE_DATA_RECEIVED

<si>
Status information (output, type X(4))
Send authorization received or not (IMS: Always set)

CM_SEND_RECEIVED
CM_NO_STATUS_RECEIVED
COMMUNICATION RECEIVE: Receive data

HOLD

If you use COMMUNICATION RECEIVE with parameter HOLD (as of R/3 Release 2.1), no roll-out and roll-in will be performed for RECEIVE. The previously requested task will be held instead.

Data length (output, type X(4))

Number of received characters in the buffer

Maximum length of the receivable buffer (input, type P)

If none is defined, the length of the buffer is used internally.

Return code (output, type X(2))

Alternatively, you can check the value directly via SY-SUBRC.

Return Codes

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_OK</td>
<td></td>
</tr>
<tr>
<td>CM_DEALLOCATED_ABEND</td>
<td>Partner has closed the connection UTM</td>
</tr>
<tr>
<td>CM_DEALLOCATED_NORMAL</td>
<td>Partner has closed the connection normally</td>
</tr>
</tbody>
</table>

DATA: CONV_ID(8) TYPE C,

BUF(255),

DI(4) TYPE X,

SI(4) TYPE X,

RL(4) TYPE X,

RC LIKE SY-SUBRC.

COMMUNICATION RECEIVE ID CONV_ID

BUFFER BUF

DATAINFO DI

STATUSINFO SI

HOLD

RECEIVED RLEN

RETURNCODE RC.

IF RC <> CM_OK....
COMMUNICATION DEALLOCATE: Close the connection

Use
A conversation [Extern] with a communications partner is closed and system resources in use are released. This statement can only be called in send status.

Activities

Syntax
COMMUNICATION DEALLOCATE ID <conv_id>

[ RETURNCODE <rc> ]

Parameter Values

<conv_id>
Conversation ID (input, type C(8))

<rc>
Return code (output, type X(2))

Return Codes

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_OK</td>
<td></td>
</tr>
<tr>
<td>CM_PROGRAM_STATE_CHECK</td>
<td>Program is not in send status</td>
</tr>
<tr>
<td>CM_PROGRAM_PARAMETER_CHECK</td>
<td>Invalid conversation ID</td>
</tr>
</tbody>
</table>

DATA: CONV_ID(8) TYPE C,
      RC LIKE SY-SUBRC.
COMMUNICATION DEALLOCATE ID CONV_ID RETURNCODE RC.
IF RC <> CM_OK....

Only the ABAP program, not the external partner program, can close the connection.
Return Codes

********************************************************************************
* Defines for ABAP CPI-C Communication
*
* Note(s):
*
* 1) ABAP supports CPI-C Starter Set only
*    Standard Values marked with "<" are used
* 2) Values are in hexadecimal representation
*
********************************************************************************

* reference fields
DATA: INT2(2) TYPE X,
       INT4(4) TYPE X.

* return_code (LIKE SY-SUBRC.
DATA: CM_OK                            LIKE INT2 VALUE '0000',
       CM_ALLOCATE_FAILURE_NO_RETRY     LIKE INT2 VALUE ['0001'],
       CM_ALLOCATE_FAILURE_RETRY        LIKE INT2 VALUE ['0002'],
       CM_CONVERSATION_TYPE_MISMATCH    LIKE INT2 VALUE ['0003'],
       CM_SECURITY_NOT_VALID            LIKE INT2 VALUE ['0006'],
       CM_SYNC_LVL_NOT_SUPPORTED_PGM    LIKE INT2 VALUE ['0008'],
       CM_TPN_NOT_RECOGNIZED            LIKE INT2 VALUE ['0009'],
       CM_TP_NOT_AVAILABLE_NO_RETRY     LIKE INT2 VALUE ['000A'],
       CM_TP_NOT_AVAILABLE_RETRY        LIKE INT2 VALUE ['000B'],
       CM_DEALLOCATED_ABEND             LIKE INT2 VALUE ['0011'],
       CM_DEALLOCATED_NORMAL            LIKE INT2 VALUE ['0012'],
       CM_PARAMETER_ERROR               LIKE INT2 VALUE ['0013'],
       CM_PRODUCT_SPECIFIC_ERROR        LIKE INT2 VALUE ['0014'],
       CM_PROGRAM_ERROR_NO_TRUNC        LIKE INT2 VALUE ['0015'],
       CM_PROGRAM_ERROR_PURGING         LIKE INT2 VALUE ['0016'],
       CM_PROGRAM_ERROR_TRUNC           LIKE INT2 VALUE ['0017'],
       CM_PROGRAM_PARAMETER_CHECK       LIKE INT2 VALUE ['0018'],
       CM_PROGRAM_STATE_CHECK           LIKE INT2 VALUE ['0019'],
       CM_ALLOCATE_FAILURE_RETRY        LIKE INT2 VALUE ['0002'],
       CM_SECURITY_NOT_VALID            LIKE INT2 VALUE ['0006'],
       CM_SYNC_LVL_NOT_SUPPORTED_PGM    LIKE INT2 VALUE ['0008'],
       CM_TPN_NOT_RECOGNIZED            LIKE INT2 VALUE ['0009'],
       CM_TP_NOT_AVAILABLE_NO_RETRY     LIKE INT2 VALUE ['000A'],
       CM_TP_NOT_AVAILABLE_RETRY        LIKE INT2 VALUE ['000B'],
       CM_DEALLOCATED_ABEND             LIKE INT2 VALUE ['0011'],
       CM_DEALLOCATED_NORMAL            LIKE INT2 VALUE ['0012'],
       CM_PARAMETER_ERROR               LIKE INT2 VALUE ['0013'],
       CM_PRODUCT_SPECIFIC_ERROR        LIKE INT2 VALUE ['0014'],
       CM_PROGRAM_ERROR_NO_TRUNC        LIKE INT2 VALUE ['0015'],
       CM_PROGRAM_ERROR_PURGING         LIKE INT2 VALUE ['0016'],
       CM_PROGRAM_ERROR_TRUNC           LIKE INT2 VALUE ['0017'],
       CM_PROGRAM_PARAMETER_CHECK       LIKE INT2 VALUE ['0018'],
       CM_PROGRAM_STATE_CHECK           LIKE INT2 VALUE ['0019'],

Return Codes

CM_RESOURCE_FAILURE_NO_RETRY LIKE INT2 VALUE '001A',
CM_RESOURCE_FAILURE_RETRY LIKE INT2 VALUE '001B',
CM_UNSUCCESSFUL LIKE INT2 VALUE '001C'.

* data_received
DATA: CM_NO_DATA_RECEIVED LIKE INT4 VALUE '00000000',
CM_DATA_RECEIVED LIKE INT4 VALUE '00000001',
CM_COMPLETE_DATA_RECEIVED LIKE INT4 VALUE '00000002',
CM_INCOMPLETE_DATA_RECEIVED LIKE INT4 VALUE '00000003'.

* status_received
DATA: CM_NO_STATUS_RECEIVED LIKE INT4 VALUE '00000000',
CM_SEND_RECEIVED LIKE INT4 VALUE '00000001',
CM_CONFIRM_RECEIVED LIKE INT4 VALUE '00000002',
CM_CONFIRM_SEND_RECEIVED LIKE INT4 VALUE '00000003',
CM_CONFIRM_DEALLOC_RECEIVED LIKE INT4 VALUE '00000004'.

* request_to_send_received
DATA: CM_REQ_TO_SEND_NOT_RECEIVED LIKE INT4 VALUE '00000000',
CM_REQ_TO_SEND_RECEIVED LIKE INT4 VALUE '00000001'.

* conversation_type
DATA: CM_BASIC_CONVERSATION LIKE INT4 VALUE '00000000',
CM_MAPPED_CONVERSATION LIKE INT4 VALUE '00000001'.

* deallocate_type
DATA: CM_DEALLOCATE_SYNC_LEVEL LIKE INT4 VALUE '00000000',
CM_DEALLOCATE_FLUSH LIKE INT4 VALUE '00000001',
CM_DEALLOCATE_CONFIRM LIKE INT4 VALUE '00000002',
CM_DEALLOCATE_ABEND LIKE INT4 VALUE '00000003'.

* error_direction
DATA: CM_RECEIVE_ERROR LIKE INT4 VALUE '00000000',
CM_SEND_ERROR LIKE INT4 VALUE '00000001'.

* fill
DATA: CM_FILL_LL LIKE INT4 VALUE '00000000',
CM_FILL_BUFFER LIKE INT4 VALUE '00000001'.

80 April 2001
* prepare_to_receive_type
DATA: CM_PREP_TO_RECEIVE_SYNC_LEVEL LIKE INT4 VALUE '00000000',
     CM_PREP_TO_RECEIVE_FLUSH LIKE INT4 VALUE '00000001',
     CM_PREP_TO_RECEIVE_CONFIRM LIKE INT4 VALUE '00000002'.

* receive_type
DATA: CM_RECEIVE_AND_WAIT LIKE INT4 VALUE '00000000',
     CM_RECEIVE_IMMEDIATE LIKE INT4 VALUE '00000001'.

* return_control
DATA: CM_WHEN_SESSION_ALLOCATED LIKE INT4 VALUE '00000000',
     CM_IMMEDIATE LIKE INT4 VALUE '00000001'.

* send_type
DATA: CM_BUFFER_DATA LIKE INT4 VALUE '00000000',
     CM_SEND_AND_FLUSH LIKE INT4 VALUE '00000001',
     CM_SEND_AND_CONFIRM LIKE INT4 VALUE '00000002',
     CM_SEND_AND_PREP_TO_RECEIVE LIKE INT4 VALUE '00000003',
     CM_SEND_AND_DEALLOCATE LIKE INT4 VALUE '00000004'.

* sync_level
DATA: CM_NONE LIKE INT4 VALUE '00000000',
     CM_CONFIRM LIKE INT4 VALUE '00000001'.

April 2001
CPI-C Interface in C

The CPI-C interface in C can be split into the following areas:

- CPI-C Development Libraries [Seite 83]
- Implemented CPI-C Function Calls [Seite 88]
- Define Variables for Computer Types [Seite 130]
- Linking an SAP Development Library [Seite 87]
CPI-C Development Libraries

Purpose
SAP provides optional platform-specific CPI-C libraries (CPI-C Subsets) for workstations, which you can use for the development of communications programs in the C language.

You can use these libraries to create "more portable" communications programs. This allows communication between SAP systems and external systems, which support CPI-C. The following communications options are available:

- R/3 ↔ C program
- R/2 ↔ C program

Features
These libraries implement the calls of the CPI-C Starter Set, parts of the Advanced Function Calls and SAP-specific calls.

The file cpic.readme contains up-to-date information on the delivered files and using the example programs. The following files are available:

- Header file for CPI-C
- Development libraries:
  - cpicslib for SNA communication, see Libraries Based on LU6.2: cpicslib [Seite 84]
  - cpictlib for TCP/IP communication, see Libraries Based on TCP/IP: cpictlib [Seite 86]
- Test programs for CPI-C communication, a calling program and a callable program:
  - in ABAP
  - in C
(See Working with the Test Programs in the documentation SAP Communication: Configuration)

- Example of a side info file
Libraries Based on LU6.2: cpicslib

**Definition**

SAP-CPI-C development libraries based on the communications protocol SNA-LU6.2 have the name `cpicslib` and a platform-specific extension.

The CPI-C functions of `cpicslib` are mapped directly to LU6.2 level.

**Use**

For communication with an R/3 System, you need a communications subsystem on both computers and the SAP Gateway on the R/3 computer.

Depending on your requirements, you can alternatively use the manufacturer-specific interface (for example, LU6.2/APPC) directly for communication with the R/2 host.

In both cases the application program communicates via the communications subsystem with the ABAP program on the R/2 host (without the SAP Gateway).
 Libraries Based on LU6.2: cpicslib
**Libraries Based on TCP/IP: cpictlib**

The respective SAP-CPI-C development library based on the communications protocol TCP/IP has the name `cpictlib` and a platform-specific extension. These libraries are based on the available TCP/IP implementations of different manufacturers. This is the library that communicates with partner programs (R/2, R/3 ABAP program, external program) via the SAP Gateway.
Linking an SAP Development Library

Prerequisites

If you use functions of one of the two SAP development libraries in your C program, link the library with your program.

If you have not specified the computer type in your program with the statement `define`, you must specify it when you compile the program. The define variables are described in Define Variables for Host Types [Seite 130].

Procedure

Under UNIX, you link an SAP development library with your program as follows:

```bash
cc -D<Define variable> <Program name> <SAP library> -o <Executable file>
```

```bash
cc -DSAPonHP_UX cpict1.c cpictlib.o -o cpict1
```

With SNC support (as of Release 3.1G) you must specify the dynamic link option on certain platforms.
Implemented CPI-C Function Calls

**Definition**

Implemented CPI-C function calls are available for programming a conversation between different programs (on different systems).

The following categories of CPI-C function calls are implemented in the libraries:

- CPI-C Starter Set [Seite 89]
- Advanced Function Calls [Seite 99]
- SAP-Specific CPI-C Functions [Seite 110]
CPI-C Starter Set

Definition

These are the basic functions required for a simple conversation.

Use

Communication between two partner programs via the calls of the CPI-C Starter Set runs as follows:

1. Establish connection (session/conversation)
   - Initialization of the communications parameters (via the side information)
   - Build a logical connection (session):
   - Start request from the remote transaction program
   - Both partners are given a conversation ID.

2. Send/receive information
   - Send data
   - Receive data/status messages

3. Close connection
   Close conversation

Structure

The CPI-C Starter Set consists of the following function calls:

<table>
<thead>
<tr>
<th>Call</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMINIT [Seite 91]</td>
<td>Initialize connection</td>
</tr>
<tr>
<td>CMALLC [Seite 94]</td>
<td>Allocate conversation</td>
</tr>
<tr>
<td>CMACCP [Seite 92]</td>
<td>Accept conversation</td>
</tr>
<tr>
<td>CMSEND [Seite 95]</td>
<td>Send data</td>
</tr>
<tr>
<td>CMRCV [Seite 96]</td>
<td>Receive data</td>
</tr>
<tr>
<td>CMDEAL [Seite 98]</td>
<td>Deallocate conversation</td>
</tr>
</tbody>
</table>

These functions are all of type CM_RETCODE.

CPI-C works in Two Way Alternate Mode (half-duplex mode).

Only one of the programs has send authorization at any one time. This authorization can be transferred to the partner (status).

The program, which builds communication, has send authorization first.

The send authorization is transferred by transmitting a receipt acknowledgement (CMRCV) in send status. Then, the partner program is authorized to send. Multiple change of communications direction is possible.
CPI-C Starter Set

The program with the send authorization can call the following functions:

- **CMSEND** Send data
- **CMRCV** from the partner
- **CMDEAL** Close the connection

A side information table with connection parameters must be available in the calling system. CMINIT needs these connection parameters.
CMINIT

The call CMINIT initializes values for a connection. A unique value (*conversation ID*) is returned to the program. This value must be specified in all subsequent calls. A symbolic address must also be specified.

**Syntax**

CMINIT ( conv_id, dest, rc)

**Parameters**

*conv_id*

Conversation ID A unique value returned by the routine.

*dest*

Symbolic name This input value must match an entry in the side info table. The entry contains parameters to build the connection.

*rc*

Return code  CM_OK : Routine was executed without errors. The return code contains the same value as rc.
CMACCP

Use
Before a program can connect with a target program, the target program must already be active and the call CMACCP must have been made. This also sets initial values. However, beforehand the logical name SAPCPICSYMDEST has to be defined that corresponds to the symbolic address for CMINIT. A unique value (conversation ID) is returned to the program. This value must be specified in all subsequent calls.

Activities

Syntax
CMACCP ( conv_id,  rc)

Parameters
conv_id
This unique value is returned by the routine.
rc
Return code (output value)
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.

Registering Programs at the SAP Gateway Without Changing the Source Code

If you specify relevant parameters when calling the target program, the program is registered at the gateway with the CMACCP call.

The CMACCP call analyzes the arguments passed. CMACCP then performs the calls SAP_CMREGTP and SAP_CMACCTP (see SAP_CMREGTP, SAP_CMACCTP, SAP_CMUNACCTP, SAP_CMUNREGTP und SAP_CMNOREGTP [Seite 123]).

This means you can use registered programs without making changes to the source code. You must link such programs with the most recent library.

Transfer parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>Transaction program name</td>
</tr>
<tr>
<td>GWHOST</td>
<td>Gateway platform</td>
</tr>
<tr>
<td>GWSERV</td>
<td>Gateway service</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Time limit in seconds (optional) If you do not specify this parameter, no timeout occurs and the program waits until the next request.</td>
</tr>
</tbody>
</table>

Calling the program:

```
cpict2  TP=cpict2  GWHOST=hs0011  GWSERV=sapgw00
```
CMALLC

Use
This call builds a connection with the partner program.

Activities

Syntax
CMALLC ( conv_id, rc)

Parameters
conv_id
The conversation ID is the input value for an initialized conversation.
rc
Return code (output value)
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
CMSEND

Use
This call sends data to the partner program. The maximum number of characters, which can be sent with one call is limited to 32000.

If a program has send authorization, it can call the following functions:

- CMSEND Send data
- CMRCV Pass send authorization and wait for data from the partner
- CMDEAL Close the connection

Activities

Syntax
CMSEND (conv_id, buffer, send_length, request_to_send_received, rc)

Parameters
conv_id
ID of the conversation, via which data is to be sent (input value)

buffer
Buffer to be sent

send_length
The number of characters to be sent (input value)

request_to_send_received
Variable indicating whether the partner wants to send data (return code). This variable can have two possible values:

- CM_REQ_TO_SEND_NOT_RECEIVED
- CM_REQ_TO_SEND_RECEIVED

rc
Return code (output value)
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
CMRCV

Use
The call CMRCV prepares a program to receive data from its partner program. If a program passes CMRCV in send mode, the program passes the send authorization to the partner program. The communications direction can be changed several times.

Activities

Syntax
CMRCV (conv_id, buffer, requested_length, data_received, received_length, status_received, request_to_send_received, rc)

Parameters

conv_id
ID of the conversation, via which data is to be received (input value)

buffer
Buffer, to which the received data is written (input value)

requested_length
Maximum number of characters that can be received (input value)

data_received
This variable (return code) indicates whether the program has received data. If the program has received data, the variable contains information on the data received. The variable can have the following values:

- CM_NO_DATA_RECEIVED
- CM_DATA_RECEIVED (nur Basic Conversation)
- CM_COMPLETE_DATA_RECEIVED
- CM_INCOMPLETE_DATA_RECEIVED

received_length
Variable containing the number of characters received (return code)

status_received
This variable indicates whether the program has received status information (return code). If the program has received status information, the variable contains information on the status of the conversation. The variable can have the following values:

- CM_NO_STATUS_RECEIVED
- CM_SEND_RECEIVED
- CM_CONFIRM_RECEIVED
- CM_CONFIRM_SEND_RECEIVED
- CM_CONFIRM_DEALLOC_RECEIVED

request_to_send_received

Variable indicating whether the partner wants to send data (return code). This variable can have two possible values:

- CM_REQ_TO_SEND_NOT_RECEIVED
- CM_REQ_TO_SEND_RECEIVED

rc

Return code (output value)

CM_OK: Routine was executed without errors.
CM_DEALLOCATE_NORMAL: Connection was closed correctly by the partner program.
The return code contains the same value as rc.
CMDEAL

Use
The call CMDEAL closes the connection with the partner program and releases system resources. This call can only be used if the local program is in send status.

Activities

Syntax
CMDEAL ( conv_id, rc)

Parameters
conv_id
ID of the conversation to be closed (input value)
rc
Return code (output value)
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
Advanced Function Calls

Definition
In addition to the calls of the CPI-C Starter Set, the following Advanced Function Calls are also available (with platform-specific restrictions):

Use
Available Advanced Function Calls

<table>
<thead>
<tr>
<th>Function</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMCFM [Seite 100]</td>
<td>Requests receipt acknowledgement from partner</td>
</tr>
<tr>
<td>CMCFMD [Seite 101]</td>
<td>Sends receipt acknowledgement to partner</td>
</tr>
<tr>
<td>CMCNVO [Seite 102]</td>
<td>Converts data buffer from ASCII to EBCDIC</td>
</tr>
<tr>
<td>CMCVNI [Seite 103]</td>
<td>Converts data buffer from EBCDIC to ASCII</td>
</tr>
<tr>
<td>CMSCSP [Seite 104]</td>
<td>Sets the Conversation Security Password</td>
</tr>
<tr>
<td>CMSCST [Seite 105]</td>
<td>Sets the Conversation Security Type</td>
</tr>
<tr>
<td>CMSCSU [Seite 106]</td>
<td>Sets the Conversation Security User ID</td>
</tr>
<tr>
<td>CMSPLN [Seite 107]</td>
<td>Sets the logical unit (LU) of the partner LU</td>
</tr>
<tr>
<td>CMSTPN [Seite 108]</td>
<td>Sets the transaction program name</td>
</tr>
<tr>
<td>CMSSL [Seite 109]</td>
<td>Sets the synchronization level</td>
</tr>
</tbody>
</table>

All functions are of type CM_RETCODE.
The syntax of the Advanced Function Calls is described in the following topics.

The functions CMCFM and CMCFMD are not available in cpictlib.
CMCNVO and CMCNVI are necessary because the R/2 System on the host only expects data in EBCDIC format, whereas the workstation generally only processes ASCII data.
CMSCST, CMSCSU, CMSCSP and CMSTPN (only for IMS) are necessary, if the LU6.2 partner system uses an external security system (for example, RACF on the host).

The standard definition of the SAP-CPI-C interface and the return codes of the individual CPI-C calls are defined in the file cpic.h.
CMCFM

Use
The call CMCFM requests the partner program to acknowledge the receipt of data.

Activities

Syntax
CMCFM (conv_id, request_to_send_received, rc)

Parameters
conv_id
ID of the conversation to be acknowledged (input value)

request_to_send_received
Variable indicating whether the partner wants to send data (return code).
This variable can have two possible values:

- CM_REQ_TO_SEND_NOT_RECEIVED
- CM_REQ_TO_SEND_RECEIVED

rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
CMCFMD

Use
The call CMCFMD sends a receipt acknowledgement to the partner.

Activities

Syntax
CMCFMD ( conv_id, rc)

Parameters
conv_id
ID of the conversation to be acknowledged (input value)

rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
CMCNVO

Use
The call CMCNVO converts a data buffer from ASCII to EBCDIC. It uses the library function LIB$TRAASCEBC.

Activities

Syntax

CMCNVO (buffer, length, rc)

Parameters

buffer
Buffer to be converted

length
Number of characters to be converted (input value)

rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.

(for UNIX)
Under UNIX, you can define your own conversion table. The SAP standard tables are contained in Conversion Tables EBCDIC ↔ ASCII [Seite 219].
CMCVNI

Use
The call CMCVNI converts a data buffer from EBCDIC to ASCII. It uses the library function LIB$TRAEBCASC.

Activities

Syntax
CMCNVI (buffer, length, rc)

Parameters
buffer
Buffer to be converted (input value)
length
Number of characters to be converted (input value)
rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.

(for UNIX)
Under UNIX, you can define your own conversion table. The SAP standard tables are contained in Conversion Tables EBCDIC <-> ASCII [Seite 219].
CMSCSP

Use
The call CMSCSP sets the password for conversation security.

Activities

Syntax
CMSCSP (conv_id, security_password, security_password_length, rc)

Parameters
conv_id
ID of the conversation, for which the password is to be set (input value)

security_password
Conversation security password (input value)

security_password_length
Length of the password (input value)
Possible values: 0 to 8

rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
CMSCST

Use
The call CMSCST sets the conversation security type.

Activities

Syntax
CMSCST (conv_id, security_type, rc)

Parameters
conv_id
ID of the conversation, for which the conversation security type is to be set (input value)

security_type
indicates which user information the program sends to its partner (input value)

Possible values are:

  • CM_SECURITY_NONE:
    Neither the user ID nor the password are sent to the partner program.

  • CM_SECURITY_SAME:
    The user ID is sent to the partner.

  • CM_SECURITY_PROGRAM:
    The user ID and the password are sent to the partner.

rc
Return code
CM_OK: Routine was executed without errors.

The return code contains the same value as rc.
CMSCSU

Use
The call CMSCSU sets the user ID for the conversation.

Activities

Syntax
CMSCSU (conv_id, user_id, user_id_length, rc)

Parameters
conv_id
ID of the conversation, for which the user ID is to be set (input value)
user_id
User ID (input value)
user_id_length
Length of the user ID (input value)
Possible values: 0 to 8
rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
**CMSPLN**

**Use**

The call CMSPLN sets the Logical Unit (LU) of the partner Logical Unit.

**Activities**

**Syntax**

CMSPLN (conv_id, partner_lu, partner_lu_len, rc)

**Parameters**

conv_id
ID of the conversation, for which the Logical Unit of the partner is to be set (input value)

partner_lu
Name of the Logical Unit of the partner system (input value)

partner_lu_len
Length of the Logical Unit (input value)
Possible values: 1 to 8.

rc
Return code
CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
CMSTPN

Use
The call CMSTPN sets the name of the remote transaction program.

Activities

Syntax
CMSTPN (conv_id, tpname, tpname_len, rc)

Parameters
conv_id
ID of the conversation, for which the name of the remote transaction program is to be set (input value)

tpname
Name of the remote transaction program (input value)

tpname_len
Length of the name

rc
Return code

CM_OK: Routine was executed without errors.
The return code contains the same value as rc.
**CMSSL**

**Use**

This call sets the synchronization level.

**Activities**

**Syntax**

CMSSL (conv_id, sync_level, rc)

**Parameters**

conv_id

ID of the conversation, for which the synchronization level is to be set (input value)

sync_level

Synchronization level to be set (input value)

Possible values are:

- CM_NONE: No synchronization
- CM_CONFIRM: Sync_level CONFIRM

rc

Return code

CM_OK: Routine was executed without errors.

The return code contains the same value as rc.
## SAP-Specific CPI-C Functions

### Definition

The SAP-specific CPI-C functions pass additional parameters to the SAP-CPI-C interface, which cannot be mapped to the standard CPI-C parameters.

### Use

The following SAP-specific CPI-C functions are provided:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Via <strong>SAP_CMINIT</strong> [Seite 111]:</td>
<td>Passes on gateway parameters</td>
</tr>
<tr>
<td><strong>SAP_CMACCP</strong> [Seite 112]</td>
<td>Passes on accept parameters</td>
</tr>
<tr>
<td><strong>SAP_CMPERR</strong> [Seite 113]:</td>
<td>Displays error information</td>
</tr>
<tr>
<td><strong>SAP_CMLOGON</strong> [Seite 114]</td>
<td>Creates logon string</td>
</tr>
<tr>
<td><strong>SAP_CMCERR</strong> [Seite 116]:</td>
<td>Identifies error information</td>
</tr>
<tr>
<td><strong>SAP_CMLOADCONVTAB</strong> [Seite 117]:</td>
<td>Loads the conversion table specified</td>
</tr>
<tr>
<td><strong>SAP_CMMODCONVTAB</strong> [Seite 118]:</td>
<td>Modifies the current conversion table</td>
</tr>
<tr>
<td><strong>SAP_CMTIMEOUT</strong> [Seite 119]:</td>
<td>Specifies a timeout value</td>
</tr>
<tr>
<td><strong>SAP_CMHANDLE</strong> [Seite 120]</td>
<td>Returns the socket handle</td>
</tr>
<tr>
<td><strong>SAP_CMGWHOST</strong> [Seite 121]:</td>
<td>Determines the gateway host</td>
</tr>
<tr>
<td><strong>SAP_CMGWSERV</strong> [Seite 122]:</td>
<td>Determines the gateway service</td>
</tr>
<tr>
<td><strong>SAP_CMREGTP</strong></td>
<td>Registers the program with the SAP-Gateway</td>
</tr>
<tr>
<td><strong>SAP_CMACCPTP</strong></td>
<td>Ready for connection setup</td>
</tr>
<tr>
<td><strong>SAP_CMUNACCTP</strong></td>
<td>No longer ready for connection setup</td>
</tr>
<tr>
<td><strong>SAP_CMUNREGTP</strong></td>
<td>Deregisters the program</td>
</tr>
<tr>
<td><strong>SAP_CMNOREGTP</strong></td>
<td>Determines the number of registered programs</td>
</tr>
<tr>
<td><strong>SAP_CMCANCREGTP</strong></td>
<td>Logs registered program off Gateway</td>
</tr>
<tr>
<td><strong>SAP_CMGETVERSION</strong></td>
<td>Returns the internal version of the CPI-C-library</td>
</tr>
<tr>
<td><strong>SAP_CMSNMCMODE</strong></td>
<td>Only relevant if you use the Secure Network</td>
</tr>
<tr>
<td><strong>SAP_CMSNMCNAME</strong></td>
<td>Communications interface to third-party security</td>
</tr>
<tr>
<td><strong>SAP_CMACLKEY</strong></td>
<td>systems</td>
</tr>
<tr>
<td><strong>SAP_CNAMETOACLKEY</strong></td>
<td>SNC status of a connection</td>
</tr>
<tr>
<td><strong>SAP_CMACLKEYTOUNAME</strong></td>
<td>(SNC_ON/SNC_OFF)</td>
</tr>
<tr>
<td><strong>SAP_CMGETVERSION</strong></td>
<td>Returns the SNC name of the partner</td>
</tr>
<tr>
<td></td>
<td>Returns the ACL key of the partner</td>
</tr>
<tr>
<td></td>
<td>Converts the SNC name to an ACL key</td>
</tr>
<tr>
<td></td>
<td>Converts the ACL key to an SNC name</td>
</tr>
</tbody>
</table>

110 April 2001
SAP_CMINIT

Use

SAP_CMINIT passes the following additional parameters to the SAP-CPI-C interface:

- Host, on which the SAP Gateway is running
- Service, to which the SAP Gateway responds
- Protocol type of the connection to be built

These values are needed to build the connection to the SAP Gateway.

The header file `cpic.h` indicates the prototypes.

In the following cases, SAP_CMINIT delivers a return code not equal to CM_OK:

- The specified service is too long.
- The specified host name is too long.
- None of the values C, I, E or G was specified for the protocol.

Integration

- You do not need to use the call SAP_CMINIT if you have defined all the parameters in the side information table.
  For reasons of compatibility, you should always define the parameters in this table.
- The above parameters are not needed for the CPI-C development library `cpicslib`, which is based on SNA.
  The function SAP_CMINIT always returns the value CM_OK in this environment.

Activities

```c
main (argv, argc)
...
SAP_CMINIT(argv, "compu01", "sapgw00", INT_SOCK_COMM);
...
```

The SAP Gateway runs on the host `compu01` and responds to the service `sapgw00`. An internal communication is built. The partner program is therefore an ABAP program in an R/3 System.
SAP_CMACCP

Use

The parameters needed in the called program to build the connection are passed as call parameters with both cpicslib and cpictlib. For this reason, the CPI-C interface needs to access the argument vector. By calling the function SAP_CMACCP, the address of the argument vector is passed to the CPI-C interface.

The header file cpic.h indicates the prototypes. SAP_CMACCP always returns the value CM_OK.

```c
main (argv, argc)
   ...
   SAP_CMACCP (argv);
   ...
```

Integration

Registering Programs at the SAP Gateway Without Changing the Source Code

If you specify relevant parameters when calling the target program, the program is registered at the gateway with the SAP_CMACCP call.

The SAP_CMACCP call analyzes the arguments passed. SAP_CMACCP then performs the calls SAP_CMREGTP and SAP_CMACCTP (see Functions for Registered CPI-C Programs [Seite 123]).

This means you can use registered programs without making changes to the source code. You must link such programs with the most recent library.

Transfer parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>Transaction program name</td>
</tr>
<tr>
<td>GWHOST</td>
<td>Gateway platform</td>
</tr>
<tr>
<td>GWSERV</td>
<td>Gateway service</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>Time limit in seconds (optional) If you do not specify this parameter, no timeout occurs and the program waits until the next request.</td>
</tr>
</tbody>
</table>

Calling the program:

```c
cpitc2 TP=cpitc2 GWHOST=hs0011 GWSERV=sapgw00
```
SAP_CMPERR

Use
If you are using the SAP-specific function SAP_CMPERR and an error occurs, a short description of the error is output.

```c
CM_RETCODE return_code;
...;
CMALLC(conv_id,&return_code);
if (return_code != CM_OK)
{
    printf("SAP-INFO: %s\n", SAP_CMPERR());
}
...;
```
SAP Communication: CPI-C Programming (BC-CST-GW)

SAP_CMLOGON

Use
The function SAP_CMLOGON builds the logon sequence (connect string), which is needed to log on to an R/2 or an R/3 System. When the connection is built (allocate), this character sequence must be the first data sent to the target system.

Features
Parameters of the function SAP_CMLOGON:

<table>
<thead>
<tr>
<th>Buffer</th>
<th>Pointer to the buffer of the calling program</th>
</tr>
</thead>
<tbody>
<tr>
<td>len</td>
<td>Length of the user buffer</td>
</tr>
<tr>
<td>reqid</td>
<td>From { &quot;CONN&quot;, &quot;FREE&quot;, &quot;APPC&quot;,.. }</td>
</tr>
<tr>
<td>reqtype</td>
<td>From { &quot;DYNP&quot;, &quot;RDIA&quot;, &quot;RODC&quot;, &quot;CPIC&quot;, &quot;GRAF&quot;,.. }</td>
</tr>
<tr>
<td>amode</td>
<td>No. of the alternative mode, from { 1,.., 6 }</td>
</tr>
<tr>
<td>mand</td>
<td>Client</td>
</tr>
<tr>
<td>name</td>
<td>User name</td>
</tr>
<tr>
<td>code</td>
<td>Password</td>
</tr>
<tr>
<td>lang</td>
<td>Logon language</td>
</tr>
<tr>
<td>prog</td>
<td>Program to be started</td>
</tr>
<tr>
<td>modn</td>
<td>Form to be started</td>
</tr>
<tr>
<td>rc</td>
<td>Return code</td>
</tr>
</tbody>
</table>

To build the connect string, you have two alternatives:

- In the calling program:
  
The calling program must specify a valid pointer as the parameter buffer, which points to a sufficiently large buffer.

- In a static area of the function SAP_CMLOGON:
  
The parameter buffer must have the value 0.

Activities

```c
CPIC_CHAR buf[200];
CPIC_INT  len;

len = sizeof(buf);

SAP_CMLOGON(buf,
            &len,
            "CONN",
```
The logon character string for user "SMITH" with the password "SECRET" in client "000" is built in the buffer *buf*. In alternative mode "1", the ABAP form "TEST" is started in program "EXAMPLE". After the call, the parameter *len* contains the length of the generated logon sequence. This character string can then be sent using CMSEND.
**SAP_CMCERR**

**Use**

SAP_CMCERR checks whether the data received from the R/2 or R/3 System constitutes an error message. If it does, the message is formatted for output and a pointer is returned to the error text. If the data does not contain an error message, SAP_CMCERR returns the value 0.

```c
PCPIC_CHAR s;

..<Build connect string and send to R/2 or R/3>

..CMRCV(conv_id, input, &requested_length, &data_received,
       &received_length, &status_received,
       &request_to_send_received, &return_code);

if ((return_code != CM_OK) &&
    (return_code != CM_DEALLOCATED_NORMAL))
{
    printf("CMRCV: %d\n",return_code);
    printf("SAP-INFO: %s\n", SAP_CMPERR());
    exit(1);
}

if (return_code == CM_DEALLOCATED_NORMAL)
{
    if ((s = SAP_CMCERR(input, &received_length)) !=
        (PCPIC_CHAR *)0)
    {
        printf("CPIC-Login-Error: %s\n", s);
        exit(1);
    }
    else
    {
        printf("CMRCV: %d\n",return_code);
        printf("SAP-INFO: %s\n", SAP_CMPERR());
        exit(1);
    }
}
..
SAP_CMLOADCONVTAB

Use

SAP_CMLOADCONVTAB loads the specified conversion table and overwrites the previous conversion table. All subsequent CMCNVI and CMCNVO calls work with the new conversion table.

```c
CM_RETCODE return_code;
.
SAP_CMLOADCONVTAB("my_conv_table", &return_code);
```
SAP COMMUNICATION: CPI-C PROGRAMMING (BC-CST-GW)

SAP Kommunikations: CPI-C Programmierung (BC-CST-GW)

SAP_CMMODCONVTAB

**SAP_CMMODCONVTAB**

**Use**

You can use SAP_CMMODCONVTAB to modify a contiguous area in the current conversion table. This allows you to change the conversion of individual characters. You can only change either the ASCII → EBCDIC table or the EBCDIC → ASCII table.

```c
CM_RETCODE return_code;
...
SAP_CMMODCONVTAB(EBCDIC_TABLE, 193, "4243", &return_code);
...
```

The above modification results in the following (not very useful) conversion:

<table>
<thead>
<tr>
<th>EBCDIC character</th>
<th>ASCII character</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

All subsequent CMCDNVI and CMCDNVO calls work with the modified conversion table.
SAP_CMTIMEOUT

Use

SAP_CMTIMEOUT controls the behaviour of the "blocking" CPI-C functions. Blocking CPI-C functions are functions, which normally only return to the caller if the return code was sent by the SAP Gateway.

This can sometimes take a long time. In environments such as WINDOWS, this would cause the whole PC to be blocked. You can use SAP_CMTIMEOUT to specify a time in milliseconds, after which the function returns. If a timeout is set, CM_SAP_TIMEOUT_RETRY is returned as a return code. The function must then be called again (see also function SAP_CMHANDLE [Seite 120]).

```c
CM_RETCODE return_code;
CPIC_INT timeout;

..
  timeout = 10;
  SAP_CMTIMEOUT(timeout, &return_code);
..```

Possible values for timeout:

- **SAP_CMBLOCK**: No timeout
  CPI-C functions wait "indefinitely" for the return code (default setting)
- **Value >= 0**: Timeout in milliseconds; The value zero caused an immediate timeout.
SAP CMHANDLE

**SAP_CMHANDLE**

**Use**

SAP_CMHANDLE returns the *socket handle*, via which the CPI-C interface reads the return codes and data from the SAP Gateway. This *handle* and the function SAP_CMTIMEOUT (see above) allow the CPI-C interface to be operated without blocking. The *handle* can, for example, be used for the function Select. This makes it possible to wait for more than one event. An event is the arrival of CPI-C data.

**Integration**

The function SAP_CMHANDLE can only be used in connection with the function CMINIT or CMACCP.

```c
CM_RETCODE return_code;
CPIC_INT gwhandle;

...

SAP_CMHANDLE(&gwhandle, &return_code);
...
```
**SAP_CMGWHOST**

**Use**
For CPI-C programs, which are started by the CPI-C interface, this function checks the argument vector to find the gateway host computer.

**Integration**
You can only use this function after calling the function SAP_CMACCP.

```
Example: CM_RETCODE return_code;
CPIC_CHAR *gwhost, *gwserv;

SAP_CMACCP(argv);
..
SAP_CMGWHOST (&gwhost,&return_code);
..
SAP_CMGWSERV (&gwserv,&return_code);
..```

SAP_COMMUNICATION_CPI-C_PROGRAMMING (BC-CST-GW)

SAP_CMGWSERV

SAP_CMGWSERV

Use

For CPI-C programs, which are started by the CPI-C interface, this function checks the argument
vector to find the gateway service.

Integration

You can only use this function after calling the function SAP_CMACCP.

```c
Example: CM RETCODE return_code;
CPIC_CHAR *gwhost, *gwserv;

SAP_CMACCP(argv);
.
SAP_CMGWHOST (&gwhost, &return_code);
.
SAP_CMGWSERV (&gwserv, &return_code);
.
```
Functions for Registered CPI-C Programs

Use
You can use these functions to write a CPI-C program which, unlike “normal” CPI-C programs, is not started after the connection is established but which starts and then waits for a connection to be established.

This means that a single run of the program can accept several connections. This is particularly useful for programs which are executed very frequently, since it avoids the repeated overhead of starting up.

Features
The following SAP-specific functions are available for registered programs:

- SAP_CMREGTP
- SAP_CMACCPTP
- SAP_CMUNACCPTP
- SAP_CMUNREGTP
- SAP_CMNOREGTP
- SAP_CMCANCREGTP

For a short description refer to the chapter SAP-Specific CPI-C Functions [Seite 110].

The process consists of the following steps:

1. Registering the program
   The SAP_CMREGTP function registers a program with a TP name with the SAP Gateway. A socket handle is then returned.
   The TP name is the name of the program that waits for a connection setup.

2. Accept connection
   The SAP_CMACCPTP function waits for the connection to be established. You can specify a timeout period for the function.
   If the function times out without establishing a connection, then it terminates with the return code CM_SAP_TIMEOUT_RETRY.

   The return code CM_DEALLOCATED_NORMAL indicates that the registered program was terminated by the Gateway. Possible causes are:
   - The Gateway itself terminated.
   - Another program executed the call SAP_CMCANCREGTP().

   If the connection is successfully established then the Conversation ID is returned, and this is used in the subsequent CPI-C communication.

   After the connection has been broken, SAP_CMACCPTP() can be called again. The program does not need to be restarted each time.
If there is data at the socket handle, then you just need to call SAP_CMACCPTP(). The SELECT function can be used to check for this.

If the program is to perform other tasks without being interrupted, you can reset the registration status using the function SAP_CMUNACCPTP. The program will then not be addressed by the gateway. Calling SAP_CMACCPTP changes the status from INIT to WAITING, and makes the program wait for a new connection setup.

3. Terminating the program

The SAP_CMUNREGTP function deregisters the program and cancels the connection to the SAP Gateway.

You can use the function SAP_CMCANCREGTP to cause the Gateway to terminate a registered program. In contrast to the function SAP_CMUNREGTP the program to be terminated does not have to be logged on the Gateway by you (SAP_CMREGTP). Instead you can stop any program.

Both functions will only terminate programs that have the status INIT or WAITING.

```c
CM_RETCODE       return_code;
CPIC_CHAR       *tpname;
CPIC_CHAR       *gwhost;
CPIC_CHAR       *gwserv;
PCONVERSATION_ID convid;
CPIC_INT         handle;
CPIC_INT         timeout;

SAP_CMREGTP(tpname,gwhost,gwserv,&handle,&return_code);
for(;;)
{
    SAP_CMACCPTP(handle,timeout,convid,&return_code);
    CMRCV(...);
    CMSEND(...);
    CMDEAL(...);
}
SAP_CMUNREGTP(handle,&return_code);
```

Note the contrast to "normal" CPI-C programs which are started after the connection has been established:

```c
main (int argv, char ** argc)

CM_RETCODE       return_code;
CONVERSATION_ID convid;

SAP_CMACCP(argv);
CMACCP(convid,&return_code);
CMRCV(...);
CMSEND(...);
CMDEAL(...);
```

Example program which resets the registration status:
Functions for Registered CPI-C Programs

SAP_CMREGTP(tpname, gwhost, gwserv, &handle, &return_code);
SAP_CMACCPTP(handle, timeout, convid, &return_code);

If ( return_code == CM_SAP_TIMEOUT_RETRY)
{
    /* timeout, some other action not */
    /* to be interrupted */
    SAP_CMUNACCPTP(handle,&return_code);
    ...
    SAP_CMACCPTP(handle, timeout, convid, &return_code);
}

Determining the Number of Programs Registered at the Gateway

You can use the function SAP_CMNOREGTP() to find out how often a program has been registered with a specific name at the SAP Gateway. The total number and the program status are returned:

- **total**: Number of registered programs (total of programs in INIT, WAITING and RUNNING status)
- **init**: Number of registrations in INIT status, which means, not waiting for tasks
- **waiting**: Number of registrations in WAIT status, which means, available for new tasks
- **running**: Number of registrations in RUNNING status, which means, currently working on tasks

```
CM_RETCODE  return_code;
CPIC_CHAR   *tpname;
CPIC_CHAR   *gwhost;
CPIC_CHAR   *gwserv;
CPIC_INT    total;
CPIC_INT    init;
CPIC_INT    waiting;
CPIC_INT    running;

SAP_CMNOREGTP(tpname,gwhost,gwserv,
              &total,&init,&waiting,&running,
              &return_code);
```

The function module GWY_GET_NO_REG_PROGRAMS provides the same functionality in the R/3 System.

Changing Statuses of Registered Programs

- **SAP_CMREGTP(2)**: INIT
- **SAP_CMACCPTP (call)**: INIT → WAITING
- **SAP_CMACCPTP (rc=timeout)**: WAITING → RUNNING
Functions for Registered CPI-C Programs

- SAP_CMACCPTP (connection setup): WAITING → RUNNING
- SAP_CMUNACCPTP: WAITING → INIT
- CMDEAL: RUNNING → INIT

Communication with a registered program requires protocol type R.
**SAP_CMCANCREGTP**

**Use**

You can use the function SAP_CMCANCREGTP() to cause the Gateway to terminate a registered program. Both functions will only terminate programs that have the status INIT or WAITING. Running programs are not affected. The return code CM_DEALLOCATED_NORMAL is sent to the registered program in the SAP_CMACCPTP call.

**Syntax**

SAP_CMCANCREGTP(tpname,gwhost,gwserv,&total,&canceled,&returncode);

**Parameters**

- **tpname** Transaction program name
- **gwhost** Gateway host
- **gwserv** Gateway service
- **total** Total no. of registrations
- **canceled** No. of terminated programs
- **returncode** Return code

**Integration**

The function module GWY_CANCEL_REG_PROGRAMS provides the same functionality in the R/3 System.
SNC Function Calls

Use

Secure communication via the CPI-C interface is provided through SAP’s SNC interface (Secure Network Communication).

SNC supports third-party security systems (such as Kerberos, SECUDE, etc.). This provides for a secure Authentication of the partners and secure data transfer.

You can activate the SNC functions via environment variables or entries in the side information file (see the side information basics in the documentation SAP Communication: Configuration [Extern]).

Features

The following SNC function calls are available:

- **SAP_CMSNCMODE**
  Returns the SNC status of a connection.
  Possible values: SNC_ON, SNC_OFF
  
  ```c
  SAP_CMSNCMODE(convid, &snc_mode, &return_code);
  ```

- **SAP_CMSNCNAME**
  Returns the SNC name of the partner that has set up the connection.

  ```c
  SAP_CMSNCNAME(convid, sncname, sncname_len, &return_code);
  ```

- **SAP_CMACLKEY**
  Returns the SNC ACL key of the partner that has set up the connection.

  ```c
  SAP_CMACLKEY(convid, aclkey, aclkey_len, &return_len, &return_code);
  ```

- **SAP_CMNAMETOACLKEY**
  Converts the SNC name to an ACL key.

  ```c
  SAP_CMNAMETOACLKEY(snclib, sncname, aclkey, len, &return_len, &return_code);
  ```

- **SAP_CMACLKEYTONAME**
  Converts the ACL key to an SNC name.

  ```c
  SAP_CMACLKEYTONAME(snclib, aclkey, aclkey_len, sncname, &return_code);
  ```

- **SAP_CMINIT3**
  Analog to SAP_CMINIT [Seite 111], but with SNC data

- **SAP_CMREGTP2**
Analog to SAP_CMREGTP, but with SNC data (see Functions for Registered CPI-C Programs [Seite 123]). Converts the ACL key to an SNC name.

SAP_CMACLKEYTONAME(snclib, aclkey, aclkey_len, sncname, &return_code);
Define Variables for Host Types

Use

The header file `cpic.h` is host-independent. So that the target-host-specific parameter type is used, you must specify the host type before including the header file.

You can specify the host type in one of two ways:

- Statement `define` in the C program (see the example below)
- Parameter specification in the compiler call

(see [Linking an SAP Development Library](Seite 87))

Features

Supported host systems:

<table>
<thead>
<tr>
<th>Define variable</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAPonBULLDPX2</td>
<td>BULL DPX/2 300 with B.O.S.</td>
</tr>
<tr>
<td>SAPonMIPS</td>
<td>DEC Station 3100 with ULTRIX</td>
</tr>
<tr>
<td>SAPonHP_UX</td>
<td>HP/9000-400 with HP-UX</td>
</tr>
<tr>
<td>SAPonHP_UX</td>
<td>HP/9000 (PA-RISC) with HP-UX</td>
</tr>
<tr>
<td>SAPonRS6000</td>
<td>IBM RS6000 AIX</td>
</tr>
<tr>
<td>SAPonAS400</td>
<td>AS/400 with OS/400</td>
</tr>
<tr>
<td>SAPonMVS</td>
<td>IMS with MVS</td>
</tr>
<tr>
<td>SAPonOS2_2x</td>
<td>PS/2 with OS/2</td>
</tr>
<tr>
<td>SAPonMX3I</td>
<td>MX 300 (INTEL) with SINIX V5.4 in System V mode</td>
</tr>
<tr>
<td>SAPonMX3N</td>
<td>MX 300 with SINIX V5.2 in System V mode</td>
</tr>
<tr>
<td>SAPonMX5I</td>
<td>MX 500 (INTEL) with SINIX V5.4 in System V mode</td>
</tr>
<tr>
<td>SAPonRM600</td>
<td>RM600 (MIPS) with SINIX V5.4 in System V mode</td>
</tr>
<tr>
<td>SAPonWX2I</td>
<td>WX 200 with SINIX Open Desktop V1.0</td>
</tr>
<tr>
<td>SAPonBS2</td>
<td>BS2000</td>
</tr>
<tr>
<td>SAPonVMS</td>
<td>OpenVMS</td>
</tr>
</tbody>
</table>

HP/9000-400 with HP-UX:

```c
#define SAPonHP-UX 1
```

...
```
#define SOCK 1
#include "cpic.h"
...
main( .. )
..
Asynchronous Data Transfer With Q-API

Asynchronous Data Transfer is based on various queue interfaces.

Queue Interface in the R/3 System [Seite 139]
Queue Interfaces in the R/2 System [Seite 148]
Queue Interface for C Programs: RFC to R/3 [Seite 171]
Data Transfer

Definition

Synchronous Data Transfer

Data is transferred directly (simultaneously, synchronously) from program to program via CPI-C communication.

Both communications partners must be available at the same time. Because the central SAP system in a typical SAP installation does not run round the clock, but the linked systems are frequently in operation 24 hours a day, asynchronous data transfer is necessary.

Synchronous data transfer has the following disadvantages:

- Transfer is not possible if the partner system or the line is not available.
- A data backlog in the receiver system causes a data backlog in the sender system. Processing in the application is delayed.
- If a connection is broken, it may be necessary to perform a recovery in both systems.

Asynchronous Data Transfer

With asynchronous, or buffered, data transfer, data is temporarily stored in a sequential queue.

Asynchronous transfer has the following advantages:

- Wait times in the sender system are avoided.
- A recovery is automatically performed in the sender system.
- Transfer need not be performed during online time. This avoids placing unnecessary load on the system and thus helps to reduce costs.

The Queue Application Programming Interface (Q-API) is an SAP interface for asynchronous data transfer.

Data is buffered sequentially and processed immediately or later by an asynchronously running process.

One possible processing method is to send data to an external partner system via CPI-C. Data units that belong together can be stored in accordance with transaction and sent to a communications partner.

Data is buffered in queues before it is transferred to the target system.

In an R/3 System, the queues are stored in a relational database. All R/3 database systems are supported.

The ABAP interface is implemented as follows:

<table>
<thead>
<tr>
<th>ABAP Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In R/3:</td>
<td>ABAP Function Modules</td>
</tr>
<tr>
<td>In R/2:</td>
<td></td>
</tr>
<tr>
<td>Release 4.3H - 4.4:</td>
<td>ABAP calls to Assembler routines</td>
</tr>
<tr>
<td>Release 5.0:</td>
<td>ABAP key words</td>
</tr>
</tbody>
</table>
Data Transfer

The C interface to R/3 is implemented as a library which uses R/3 function modules through a Remote Function Call (RFC).

Integration

The following topics explain the basic terms and concepts of the SAP interface Q-API.

- Queues [Seite 135]
- Queue Attributes [Seite 136]
- Queue Element [Seite 137]
- Queue Unit [Seite 138]
Queues

Definition

Data elements from an application program are stored for transfer in a queue.

A queue is identified uniquely by a name, which is either defined by the application program or assigned automatically by the queue administration (time stamp) if you do not specify a name explicitly.

The attributes of a queue are determined when the queue is first opened via Queue Attributes [Seite 136].

Integration

Two tables are relevant for internal database administration.

- Table APQI
  This table is used as a queue directory. Each queue has one table entry, which contains its attributes and administration data.

- Table APQD
  This table is used as a data pool, in which the actual data objects are stored.
Queue Attributes

Definition
The attributes and parameters of a queue are set by the queue attributes when it is opened for the first time in write mode. The queue directory entry is created automatically.

Structure
Within an R/3 System, the queue parameters can be displayed and administered via queue administration (SM38).

- Queue name
  A maximum of 20 characters or a time stamp
- Can be described: Unique or multiple/parallel
- Data object and processing type recognition
- SAP-specific parameters for the logon protocol
  - Client
  - User password
  - ABAP program/form
- Processing parameters
  - Program
  - Start mode
  - Date/time
  - User password for Express Mail
Queue Element

Definition
A queue element (block) consists of a fixed header for queue administration and a variable data section.

```
Header    Variable data section
```

Application data are stored transparently in the variable data section, so that any data objects can be stored in the queue.

Segmenting is supported for data structures, which exceed the maximum length of the variable data section. This data is stored, in a form that the application can recognize, divided into several logical storage blocks, and passed on complete to the application when it is called again. The application program makes available a sufficiently large receiver buffer.

Note on ABAP:
- The maximum data object size is defined via the Data Dictionary field APQD-VARDATA.
- The maximum buffer size is restricted to the ABAP data range.
Queue Unit

Definition

If several data elements belong to the same processing unit, this unit is known as a transaction-oriented data unit or Queue unit.

Each relevant data element must retain a specific sequence ID:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_SINGLE (S)</td>
<td>Individual element</td>
</tr>
<tr>
<td>Q_FIRST (F)</td>
<td>First element</td>
</tr>
<tr>
<td>Q_MIDDLE ()</td>
<td>Not first/last element</td>
</tr>
<tr>
<td>Q_LAST (L)</td>
<td>Last element</td>
</tr>
</tbody>
</table>

A processing program, which is active at a later time, can only delete the data in a queue unit, if the last data object has been acknowledged as processed.
Queue Interface in the R/3 System

Definition

The R/3 System provides ABAP function modules as an interface for asynchronous data transfer. Before it is transferred, the data is stored temporarily in a relational database in queues.

As of R/3 Release 3.0 you can use the transactional RFC for buffered data transfer. For more information, refer to the documentation Remote Communications [Extern].

Use

Depending on how you use the parameters, the transfer program starts automatically or must be started manually via System → Services → Queue (transaction code SM38). Transaction code SM38 administers and displays queues and their processing logs.

A driver program transfers data at a particular time. The name of the driver program is RSQAPI20.

You must include the file RSQAPIDF in your program. It contains data definitions which enhance the readability of ABAP queue transfer programs.

The sample program for R/3 [Seite 205] shows how data is written in a queue.

Structure

You can use the following function modules with their relevant parameter values.

<table>
<thead>
<tr>
<th>ABAP Function Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE_OPEN [Seite 140]</td>
<td>Open a queue</td>
</tr>
<tr>
<td>QUEUE_PUT [Seite 142]</td>
<td>Place data in a queue</td>
</tr>
<tr>
<td>QUEUE_GET [Seite 143]</td>
<td>Read data from a queue</td>
</tr>
<tr>
<td>QUEUE_DELETE [Seite 146]</td>
<td>Delete data from a queue</td>
</tr>
<tr>
<td>QUEUE_CLOSE [Seite 144]</td>
<td>Closing a queue</td>
</tr>
<tr>
<td>QUEUE_ERASE [Seite 145]</td>
<td>Delete queue</td>
</tr>
<tr>
<td>QUEUE_SCHEDULE [Seite 147]</td>
<td>Schedule queue processing</td>
</tr>
</tbody>
</table>
QUEUE_OPEN

Use
To open a queue, call the function module QUEUE_OPEN with a sequence of EXPORTING parameters:

```
CALL FUNCTION 'QUEUE_OPEN'
  EXPORTING
    NAME  =  name
    TYPE  =  type
    ...
```

Parameters
Assign a value to each of the following parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>C(20)</td>
<td>Queue name</td>
</tr>
<tr>
<td>TYPE</td>
<td>C(1)</td>
<td>Queue attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' ': Unique: A new queue will be generated, even if a queue with the same name already exists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A : Appendable: An appendable queue is generated or opened (default)</td>
</tr>
<tr>
<td>OPENMODE</td>
<td>C(1)</td>
<td>Queue mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W : Open a queue to write (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R : Open a queue to read; explicit delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D : Open a queue to read and delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O : Open a queue to read only</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>C(8)</td>
<td>Symbolic name of the target system (Table TXCOM)</td>
</tr>
<tr>
<td>DATATYPE</td>
<td>C(4)</td>
<td>Data type/processing log:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XTAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' ' (binary)</td>
</tr>
<tr>
<td>CLIENT</td>
<td>C(3)</td>
<td>Client in the SAP target system</td>
</tr>
<tr>
<td>USERID</td>
<td>C(12)</td>
<td>User in the SAP target system</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>C(8)</td>
<td>User password</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>C(8)</td>
<td>Program in the SAP target system</td>
</tr>
<tr>
<td>FORM</td>
<td>C(30)</td>
<td>Form routine in the ABAP partner program</td>
</tr>
<tr>
<td>DRIVER</td>
<td>C(8)</td>
<td>Name of the driver program</td>
</tr>
</tbody>
</table>
### QUEUE_OPEN

<table>
<thead>
<tr>
<th>Field</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>C(1)</td>
<td>Start mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A: Automatic (in Online mode only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M: Manual</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E: Event-oriented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P: Periodically event-oriented</td>
</tr>
<tr>
<td>DATE</td>
<td>D(8)</td>
<td>Queue processing date</td>
</tr>
<tr>
<td>TIME</td>
<td>T(6)</td>
<td>Processing time</td>
</tr>
<tr>
<td>ERASE</td>
<td>C(1)</td>
<td>Delete ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you set a character other than a blank, the queue is deleted after processing.</td>
</tr>
<tr>
<td>QSTATE</td>
<td>C(1)</td>
<td>Queue status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* : No status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C: Queue will be created</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F: Queue has been finished</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E: Queue error</td>
</tr>
</tbody>
</table>

If you do not specify a name for a queue, you can retrieve a name by extending the call to the function module QUEUE_OPEN.

```plaintext
CALL FUNCTION 'QUEUE_OPEN'

EXPORTING...

IMPORTING NAME  = QUEUE.
```
**QUEUE_PUT**

**Use**

To write to a queue, call the function module QUEUE_PUT with a sequence of EXPORTING parameters:

```plaintext
CALL FUNCTION 'QUEUE_PUT'
  EXPORTING
    NAME   =  name
    STATE  =  state
    LENGTH =  length
    BUFFER =  buffer.
```

**Parameters**

Assign a value to each of the following parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>C(20)</td>
<td>Name of the queue to write to</td>
</tr>
<tr>
<td>STATE</td>
<td>C(1)</td>
<td>Start/end of a Queue unit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S: Only a single element is written to the queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F: First element in the queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Element is not the first, only or last</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L: Last element in the queue</td>
</tr>
<tr>
<td>LENGTH</td>
<td>I(4)</td>
<td>No. of bytes in the buffer to pass</td>
</tr>
<tr>
<td>BUFFER</td>
<td></td>
<td>Buffer to pass (like DDIC field APQD-VARDATA)</td>
</tr>
</tbody>
</table>
QUEUE_GET

Use

To read data from a queue, you must first call the function module QUEUE_GET with a sequence of parameter. The data buffer must have the same type and length as the DDIC field APQD-VARDATA. Larger data objects cannot currently be imported using this module.

FUNCTION 'QUEUE_GET'

EXPORTING
  NAME = Queue Name
  OPENMODE= Open Mode
  UNIT = Queue Unit
  POS = Queue Element

IMPORTING
  BUFFER = User Data
  LENGTH = Length of Data
  STATE = State of Unit/Message
  UNIT = Unit
  POS = Element

EXCEPTIONS
  EOQ
  BUFFER_ERROR
  INVALID_PARAMETER
  MEMORY_ERROR
  Q_ERROR
  SQL_ERROR.
Use

To close a queue, call the function module QUEUE_CLOSE with the EXPORTING parameter NAME.

```
CALL FUNCTION 'QUEUE_CLOSE'
    EXPORTING
        NAME = name
        OPENMODE = 'W'.
```
QUEUE_ERASE

Use

You delete a queue by calling the QUEUE_ERASE function module with the EXPORTING parameter NAME.

```
CALL FUNCTION 'QUEUE_ERASE'
      EXPORTING NAME = name.
```
USE

You can use the function module QUEUE_DELETE to delete data (unit) from a queue. This is done typically after processing is complete. End the call for this module with COMMIT WORK.

    CALL FUNCTION 'QUEUE_DELETE'
    EXPORTING
      NAME  = Name of Queue
      UNIT  = Unit
    EXCEPTIONS
      INVALID_PARAMETER
      Q_ERROR
      SQL_ERR.
    COMMIT WORK.
QUEUE_SCHEDULE

Use

You can use the function module QUEUE_SCHEDULE to schedule the queue processing program in R/3 background processing. This module is called internally at QUEUE_CLOSE, if you select the processing mode Q_AUTOSTART and if the queue has been opened in write mode.

Automatic processing of a queue is handled by scheduling the assigned ABAP program (driver) in the R/3 background job scheduling system.

Start mode A is not possible from an update task. Here, you must call the function module explicitly from the dialog task.

CALL FUNCTION 'QUEUE_SCHEDULE'

EXPORTING

  NAME      = Name of Queue
  PROGRAM   = Name of processing Program
  START     = Start Mode
  DATE      = Date of Processing
  TIME      = Time of processing
  COMMITX   = implicit Commit of Logical Unit of Work (LUW)

EXCEPTIONS

  INVALID_PARAMETER
  SCHEDULE_PROBLEM.

Integration

Queue processing

Automatic queue processing is handled by scheduling of the assigned ABAP program (driver) in the R/3 background scheduling system (transaction code SM36), where it can be monitored. Messages are logged in the background processing environment.

You can call a queue processing program manually via queue administration or directly.

For event-oriented processing, SAP_QEVENT is internally triggered by event parameter Name of Queue.

A scheduled batch job is required which starts a variant of the processing program and waits for this event.

These actions are largely performed automatically when creating a queue via the queue transaction SM38.
Queue Interfaces in the R/2 System

Definition

On the R/2 host, the queues are stored in a file called DOUT.

Q-API is integrated into the ABAP language and available in the Assembler environment.

Depending on what parameters you set for the Q-API functions, the transfer program is started automatically or must be started manually using the SAP transaction TMQD.

Use

For queue administration, you can use the following transactions:

**TMQD:**

Use this transaction (Queue display) to display an overview of the queues available in the system and start asynchronous transfer programs with a manual start ID.

**TMQM:**

This transaction (Queue maintenance) is for the system administrator and should only be available to a limited group. You can use this transaction to change and delete queues.

A queue is treated as a file. It can be opened, written to and closed:

- Opening a queue
  
  When a queue is opened, the queue attributes are set.

- Writing to a queue
  
  An important feature of the queue write function is that control is returned to the executing program without data transfer having been required to take place. This allows data to be passed to queues from the SAP system online and in update mode.

- Closing a queue
  
  When all the data is entered in the queue, the queue is closed.

Asynchronous data transfer is implemented in different ways in R/2 Systems. For more details, refer to the following sections:

- [Queue Interface for Release 5.0][1]
- [Queue Interface for Release 4.3H / 4.4][2]
- [Special Features in BS2000][3]
- [Error Messages of the SAP Transfer Program][4]
- [SAP ACCOUNTING Interface SAPSTEC (as of Release 4.3J, 4.4C and 5.0A)][5]

[1]: Seite 149
[2]: Seite 156
[3]: Seite 191
[4]: Seite 166
Queue Interface for Release 5.0

Definition

In R/2 Release 5.0, some functions in the Q-API library are implemented by specific ABAP key words.

For more information, refer to the following topics:

- ABAP Key Words [Seite 150]
- Queue Parameters [Seite 151]
- ABAP Statements [Seite 155]
- Sample Programs for R/2 Release 5.0 [Seite 208]

The ABAP sample programs contain functions and parameters which are dependent on the transfer data type.

- Data transfer in RODC format
- Data transfer from the SAP spool in CPIC format
- Data transfer in CPIC format
ABAP Key Words

The ABAP program interface for asynchronous communication via queues (DOUT queues) contains key words (functions) to open, close and write to DOUT queues.

- OPEN QUEUE <qparm>,
- TRANSFER <record> TO QUEUE <qparm>,
- CLOSE QUEUE <qparm>
Queue Parameters

Definition

Before you perform the functions, you must fill the queue parameter string <qparm>. The structure of <qparm> is defined in the Data Dictionary as a table without database QPARAM.

Use

The queue parameter string can be considered as a reference to an open queue (command OPEN QUEUE) in the buffer file DOUT. If you want to write to several DOUT queues, you can select which queue to write to.

Structure

The parameter string <qparm> consists of the following groups:

- Parameters for the Queue Name
- Parameters for the Transfer Program
- SAP-specific Parameters

The following topics list these parameters.

Parameters for the Queue Name

The following parameters determine the queue name:

- **QDEST CHAR(8):**
  Target system, for which you want to build the CPIC connection.

- **QAPPL CHAR(8):**
  Transaction program in the target system, which is started via the CPI-C connection (for example, the CICS transaction X1SA if the target system is an R/2 System).

- **QDTYP CHAR(4):**
  The types of data to be transferred.
  The transfer program delivered by SAP supports the following data types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RODC</td>
<td>data is in Remote ODC format.</td>
</tr>
<tr>
<td>CPIC</td>
<td>data is unformatted and user-defined.</td>
</tr>
<tr>
<td>SPLD</td>
<td>Corresponds to the CPIC format where data is stored intermediately in an SAP spool file instead of the queue.</td>
</tr>
</tbody>
</table>

- **QMAND CHAR(3):**
  Client in the target system
  Use this parameter only if the target system is an SAP system.

- **QABAP CHAR(8):**
  Name of the ABAP/4 program, in which the form routine to be performed is defined.
Queue Parameters

Use this parameter only if the target system is an SAP system and the transfer data type is CPIC or SPLD.

- QMODU CHAR(30):
  Name of the form routine to be performed.
  Use this parameter only if the target system is an SAP system and the transfer data type is CPIC or SPLD.

Parameters for the Transfer Program

The following parameters control the transfer program Y1SA.

- QMODE CHAR(1):
  Queue open mode:
  I: open queue for input
  O: open queue for output (default)
  U: open queue for update
  If you do not specify an open mode (blank), the queue is opened for output (default setting).
  Currently only this default setting is supported.

- QSTRT CHAR(1):
  Start mode of the transfer programs:
  A: Automatic
    The transfer program is started after each queue unit. The queue unit is controlled with the parameters QFIRS and QLAST.
  M: Manual
    You start the transfer program interactively using the start function of the SAP transaction TMQD or TMQM.

- QCORR CHAR(1):
  Responsibility for correction of a transfer error:
  S: Sender
  R: Receiver
  Currently only receiver correction responsibility is supported.

- QUPTA CHAR(1):
  Synchronous/asynchronous update in the SAP system
  S: Synchronous
  A: Asynchronous
  The values are only permitted if the target system is an SAP system and data type RODC is used.

- QSTDA DATE(8):
Start date of the transfer programs
for automatic start in format "YYYYMMDD"

- QSTTI TIME(6):
  Start time of the transfer programs
  for automatic start in format "HHMMSS"

You can start the transfer program manually or automatically as well as at a specified
start date and time. If the specified start time is before the automatic start time, the SAP
transfer program is started immediately.

- QFIRS CHAR(1):
  ID for the first record in a queue unit.

- QLAST CHAR(1):
  ID for the last record in a queue unit.

Transfer data can be grouped into queue units.
The transfer data in a queue unit is only deleted from the
queue after all the transfer data in the queue unit
has been sent and acknowledged.

If an error occurs, transfer begins with the first unit of the queue unit. A queue unit is
identified by the values of the two parameter fields QFIRS, QLAST:

<table>
<thead>
<tr>
<th>QFIRS</th>
<th>QLAST</th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>First transfer record</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Middle transfer record</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Last transfer record</td>
<td></td>
</tr>
</tbody>
</table>

You must ensure that the queue unit control entries are always closed correctly (with
QLAST). An incorrectly identified queue unit will result in data being transferred
incorrectly several times.

- QTREI CHAR(8):
  Name of the transfer program

If no name is specified (blank), the name of the SAP transfer program is automatically
used (for example, in CICS environment: Y1SA, in a BS2000 environment RSDRIVER).

If you specify a name, it will be interpreted as the name of a user-defined transfer
program.

User-defined transfer programs can be started automatically or manually.

**SAP-specific Parameters**

It is a good idea to use the SAP-specific parameters if the target system is an SAP system.

- QDUSR CHAR(12):
  User ID in the SAP system.

Enter a user name here in accordance with the data type (RODC/CPIC).

- QDPAS CHAR(8):
Queue Parameters

Password for user ID in the SAP system.

The following parameters can be used if the data type RODC is also used.

- **QDTRC** CHAR(4):
  - SAP transaction code
  - Transaction code of the SAP transaction to be started via RODC (for example, TS02).

- **QDPGM** CHAR(8):
  - Program name of the corresponding transaction
  - (for example, SAPPG02).

- **QDDYN** CHAR(4):
  - Screen number of the corresponding transaction
  - (for example, 0041, the first screen in transaction TS02)
ABAP Statements

Use

The following ABAP statements are relevant for data transfer:

- **COMMIT WORK**
  This statement is used to save all database changes. It corresponds to the Q-API function QCOMMIT. The transfer data is placed, related to transaction, in the queue as a *queue unit*.

- **ROLLBACK WORK**
  The function ROLLBACK WORK resets a queue transaction. It corresponds to the Q-API function QROLLBACK.

  If a system failure occurs, the data elements of a *queue unit*, which were not closed with COMMIT WORK, are removed from the database and database locks are reset. This ensures that a queue can never contain incomplete *queue units*. 
Transfer in BS2000

Definition
In a BS2000 environment the driver program is an ABAP program named RSDRIVER. In the following, it is referred to as the driver. The driver can only work in a running SAP online system. An external driver is not available.

Use
You can start the driver in one of the following ways:
1. Automatically after the message queue has been created
2. Manually via the administrative transactions TMQD and TMQM
3. Manually by calling up the program in dialog
4. Remotely from another application (via a free CPI-C connection)

UTM Special Features
In cases 1 and 2 processing is asynchronous, while in cases 3 and 4 it is synchronous. Asynchronous processing requires special programming, because distributed transaction processing within an asynchronous UTM transaction is possible only with UTM V3.3/UTM-D V2.0.
With earlier versions, distributed transaction processing can be simulated with asynchronous messages.

See also:
BS2000 R/2 Host: UTM-UTM Connection

Structure
You will find more information in the following topics:

- Asynchronous Driver Communication [Seite 157]
- Extensions [Seite 164]
- Notes on Installation [Seite 163]
- Queue Transfer Without Buffering [Seite 162]
Asynchronous Driver Communication

Purpose

For transfer in BS2000 [Seite 156] you can start the driver in such a way that processing runs asynchronously. This has the advantage of avoiding blocking.

Two cases can be identified:

- **Recipient is not an SAP System [Seite 160]**
- **Recipient is an SAP System [Seite 161]**

Process flow

Each communication message from the driver is initiated by a special header message, which must be read with a separate FGET. This header is to be kept and sent unchanged as a separate message (the first message) when responding to the SAP driver (FPUT header).

The driver report begins communication with an SAP logon message (CONN message) when logging on to another SAP system. If the application addressed is not a SAP system, then this logon must be acknowledged by the message 'APPCCPIC1 '. This logon can be suppressed from 5.0E if user and password are not specified when creating the queue.

Then the driver sends the first message of the first LUW (that has not yet been completely dispatched) and waits for an acknowledgement of receipt. Each message received with a length greater than zero is taken to be an acknowledgement. However, it has become customary to use the character string 'OK' for an acknowledgement. If there are other messages, the driver now sends the next message. When all the messages of an LUW have been sent and individually acknowledged by the partner system, the LUW in the file DOUT is deleted. If there is another LUW available, transfer is continued with the first data record of the next LUW.

The following diagram illustrates the communication process asynchronous driver - receiving program.
Asynchronous Driver Communication

SAP-System

INIT
FGET startinfo
FGET queuename
APRO AM lpap, ltac
FPUT header
FPUT 'CONNCPIC1...'
PEND FI

External System

INIT
FGET header
FGET 'CONNCPIC1...'
APRO AM lpap, ltac
FPUT header
FPUT 'APPCCPIC1...'
PEND FI

INIT
FGET header
FGET APPCCPIC1
APRO AM lpap, ltac
FPUT header
FPUT first data record
PEND FI

INIT
FGET header
FGET 'OK'
APRO AM lpap, ltac
FPUT header
FPUT second data record
PEND FI

INIT
FGET header
FGET 'OK'
APRO AM lpap, ltac
FPUT header
FPUT last data record
PEND FI

INIT
FGET header
FGET second data record
Processing data record
APRO AM lpap, ltac
FPUT header
FPUT 'CONNCPIC1...'
PEND FI

INIT
FGET header
FGET last data record
Processing data record
APRO AM lpap, ltac
FPUT header
FPUT 'OK'
PEND FI

INIT
FGET header
FGET first data record
Processing data record
APRO AM lpap, ltac
FPUT header
FPUT 'CONNCPIC1...'
PEND FI

INIT
FGET header
FGET 'FREECPIC1'
PEND FI

INIT
FGET header
FGET 'FREECPIC'
PEND FI
Recipient is not an SAP System

Purpose

If the receiving application is not an SAP System, the asynchronous driver requires an additional message header containing address information. The SAP APC protocol must also be observed.

Process flow

Message Header Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Type/Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTMDAIND</td>
<td>C8</td>
<td>Indicator for message header (set by the SAP-System)</td>
</tr>
<tr>
<td>UTMDPAPPL</td>
<td>C8</td>
<td>Application name of the SAP System (set by the SAP-System)</td>
</tr>
<tr>
<td>UTMDPPROC</td>
<td>C8</td>
<td>Processor name of the SAP application (set by the SAP-System)</td>
</tr>
<tr>
<td>UTMDPCNVD</td>
<td>X4</td>
<td>CONVID of the driver in the SAP System (set by the SAP-System)</td>
</tr>
<tr>
<td>UTMDSAPPL</td>
<td>C8</td>
<td>Application name of the partner application (set only if the SAP System is also the receiving system, otherwise not used)</td>
</tr>
<tr>
<td>UTMDSPROC</td>
<td>C8</td>
<td>Processor name of the partner application (set only if the SAP System is also the receiving system, otherwise not used)</td>
</tr>
<tr>
<td>UTMDCSNVD</td>
<td>X4</td>
<td>CONVID in the receiving system (set only if the SAP System is also the receiving system, otherwise not used)</td>
</tr>
</tbody>
</table>

If the partner application receives a message with this message header from the SAP driver, it must use this header at the beginning of each APC confirmation message.

The message header is not part of the driver message. It must be removed before any further processing is made. Therefore, the message header is sent as a separate UTM partial message. The first FGET receives the message header, while the second FGET receives the driver message.
Recipient is not an SAP System

Purpose
If an SAP System is the recipient of an APPQ transfer, you must make an entry in the XCOM table for the argument UTMPPROC,UTMPAPPL. Thus, the sender can be addressed (as PLU in this case) via its LPAP and LTAC names.

Prerequisites
The driver can only transfer an LUW (Logical Unit of Work) correctly if both systems remain active until the driver stops.

Process flow
You cannot continue with a transfer in a later session. If you do not observe this, you may receive confirmation messages from the partner by a driver that is no longer active. These messages cannot be addressed and thus are lost. As a DOUT LUW is considered as transferred only after all messages have been confirmed, the LUW continues to exist. If the driver for this queue is started again, the LUW is transferred again beginning with the first element. The recipient has to take this into account.

You should also consider that a check is made for a related XCOM entry as soon as a queue is opened. They key fields are the fields QPARM-QDEST (R/2 Release 5.0) and the corresponding field of the NAME Parameter of the ABAP/4 call using CALL (Releases 4.3 and 4.4).
Queue Transfer Without Buffering

If you are using UTM-D, you can send messages via Q-API to other UTM applications without buffering them in the DOUT file.

UTM-D makes sure that a message is received once only. An LUW is sent as a series of UTM partial messages of an asynchronous UTM-D transaction.

The partner application is addressed via the LPAP and LTAC names contained in the XCOM table. To choose this procedure, set the communication type (CType) in the XCOM table to D.

- You cannot use this procedure in a separate update task using Q-API, as this task is not a UTM task.
- If table XCOM exists in several clients (not recommendable in BS2000), all entries not equal to 0 in one client must also exist in Client 0.
Notes on Installation

In Client 0, you must create an SAP user record as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>RSDRIVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>DOUT</td>
</tr>
<tr>
<td>Transaction authorization</td>
<td>TM39</td>
</tr>
</tbody>
</table>

If spool files are to be sent from SPOOL, the same user must exist in all clients in which there are spool files. This is due to the fact that the driver is started in the corresponding client.
Extensions

Definition

Extensions are available for the following R/2 Releases:

- Release 5.0D
- Release 5.0E
- Release 5.0F
- Release 5.0G

These extensions are described in the following topics.

Release 5.0D

If you are using UTM Version 3.3, your system assumes that you are also using UTM-D V2.0. In this case, the driver operates synchronously without the additional protocol.

If you do not want to use the synchronous procedure due to the changes involved, you can keep the asynchronous procedure by stating communication type A in the XCOM table.

On system startup, any activity indicators still existing in the queues are reset. Queues are deleted if all of their LUWs have been transferred. New driver processes are started for queues that were started automatically and still contain LUWs to be transferred. These actions are performed before the APLZ restart.

Normally, a driver process stops as soon as the all queue elements have been transferred. However, queue elements are often inserted at the end of the queue while queue elements at the beginning are read out by the driver and sent to the recipient at the same time.

If the transfer is not continuous, it often happens that there are no more queue elements in the queue and the driver process stops. Fractions of a second later, new elements are placed in the queue. A new driver process has to started. This requires considerable time and resources (setting up a connection, creating a mode, performing user authorization checks, providing ABAP runtime environment, etc.).

It may be more efficient to not letting the driver stop as a queue gets empty but to inform the partner program about the state of the queue. The partner program can go into a waiting state and then try to receive more queue elements. In this processing mode, waiting only occurs with the partner program.

To change the driver to this procedure, you must set the NONSTOP variable to 1.

You can use the non-stop procedure only if you start the driver remotely. Message SU031 reports that the queue is empty.

Release 5.0E

If a connection or a driver process abnormally ends, the queue activity indicator is automatically reset. No manual intervention with Transaction TMQM is required. The partner application can thus continue reading out elements from the queue at a later point of time.
Release 5.0F
Asynchronous communication can be set up from an SAP report to an UTM partner, and messages can be sent. Here, the report only makes an ALLOCATE call and any SEND calls required. RECEIVE calls, of course, are not permitted.

Release 5.0G
You can use the communications type in table XCOM to control whether the UTM transaction is to be exited by SAP when communication is started (PEND RE is the standard) or to be kept open (PEND KP). The communications type K controls PEND KP. Keeping the transaction open has advantages with regard to error handling.

If you are using UTM-D V2.0, synchronous communication can be started even when the SAP transaction already has the long-running status. To do this, you simply have to assign transaction code Y2SA to transaction class 10. If the DOUT driver is also working synchronously, you must also assign Y1SA to class 10.

To disentangle asynchronous driver communication from general asynchronous communication (which can considerably hinder driver processes), an additional transaction code (Y3SA) and a new class (14) were defined for general asynchronous communication.
SAP ACCOUNTING Interface SAPSTEC (as of Release 4.3J, 4.4C and 5.0A)

Definition

ACCOUNTING data is transferred via the SAP ACCOUNTING interface SAPSTEC. The SAP transfer program writes an ACCOUNTING record before it terminates.

The two topics below cover the following subjects:

- Data structure
  - In which structure is data passed to the SAPSTEC interface?
- Fields
  - What meaning and value ranges do the fields have?

Structure

Data structure

***************************************************************
* S T C D U M M Y                                      *
*                                                             *
*        THIS DUMMYSECTION DESCRIBES THE LAYOUT OF THE        *
*        S A P - STATISTIC RECORD IN CICS ENVIRONMENT.       *
*                                                             *
***************************************************************

SPACE 1
RAPPL OPSYS=BS2
AIF (&APPLOK).$STCD01

DFHEISTG DSECT
.$STCD01 ANOP
STCBEG DS 0A
STCLEN DS H LENGTH OF RECORD
       DS XL2 RESERVED (V RECORD)
STCTASK DS CL2 SAP TASK TYPE:
       * D1, D2,... DIALOG
       * V1, V2,... POSTING TASK
       * N1, N2,... ODC TASK
       * L1, L2,... BACKGROUND TASK
       * X1, X2,... LU6.2 COMMUNICATION
       * S1, SS SPOOL TASK
STCFLAG1 DS X FLAG BYTE 1
STCIDIA EQU X'80' RECORD OF DIALOG TASK
STCIVBT EQU X'40' RECORD OF VB-TASK
STCISPL EQU X'20' RECORD OF SPOOL TASK
STCISYS EQU X'10' SYSTEM RECORD
STCIBDC EQU X'08' BATCH INPUT
STCISCD EQU X'04' SCHEDULED TRANSACTION
STCMANDT DS XL1 MANDANT
STCDATE DS PL4 DATE (PACKED), ZCSADATP
**SAP ACCOUNTING Interface SAPSTEC (as of Release 4.3J, 4.4C and 5.0A)**

<table>
<thead>
<tr>
<th>Data Description</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STCTIME</td>
<td>DS PLA</td>
<td>End time of transaction (packed)</td>
</tr>
<tr>
<td>STCTIMTR</td>
<td>DS PLA</td>
<td>Start time of transaction (EIB, PACK)</td>
</tr>
<tr>
<td>STCRESP</td>
<td>DS XL4</td>
<td>Response time of transaction (MS)</td>
</tr>
<tr>
<td>STCCTIM</td>
<td>DS XL4</td>
<td>CPU time of transaction (BS2000 ONLY)</td>
</tr>
<tr>
<td>STCATIM</td>
<td>DS XL4</td>
<td>Time, task work area is used (MS)</td>
</tr>
<tr>
<td>STCTCODE</td>
<td>DS CL4</td>
<td>SAP transaction code</td>
</tr>
<tr>
<td>STCREPID</td>
<td>DS CL8</td>
<td>If ABAP: report ID</td>
</tr>
<tr>
<td>STCLTERM</td>
<td>DS CL8</td>
<td>Terminal ID</td>
</tr>
<tr>
<td>STCACCT</td>
<td>DS CL12</td>
<td>Account number</td>
</tr>
</tbody>
</table>

*........CALL STATISTICS, NUMBER OF CALLS.....................*

<table>
<thead>
<tr>
<th>Data Description</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STCREADU</td>
<td>DS F</td>
<td>Read for update</td>
</tr>
<tr>
<td>STCREAD</td>
<td>DS F</td>
<td>Read</td>
</tr>
<tr>
<td>STCSET</td>
<td>DS F</td>
<td>Set</td>
</tr>
<tr>
<td>STCGET</td>
<td>DS F</td>
<td>Get</td>
</tr>
<tr>
<td>STCWRITE</td>
<td>DS F</td>
<td>Rewrite</td>
</tr>
<tr>
<td>STCINSRT</td>
<td>DS F</td>
<td>Insert</td>
</tr>
<tr>
<td>STCDELET</td>
<td>DS F</td>
<td>Delete</td>
</tr>
<tr>
<td>STCLOAD</td>
<td>DS F</td>
<td>PGM load</td>
</tr>
<tr>
<td>STCSYNC</td>
<td>DS F</td>
<td>Syncpoint</td>
</tr>
<tr>
<td>STCROLLB</td>
<td>DS F</td>
<td>Rollback</td>
</tr>
<tr>
<td>STCCALL</td>
<td>DS F</td>
<td>Other calls</td>
</tr>
</tbody>
</table>

*........TIMER, I/O COUNTER....................................*

<table>
<thead>
<tr>
<th>Data Description</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STCEXDA</td>
<td>DS H</td>
<td>Data excep's</td>
</tr>
<tr>
<td>STCEXIN</td>
<td>DS H</td>
<td>Index excep's</td>
</tr>
<tr>
<td>STCEXES</td>
<td>DS H</td>
<td>ESDS excep's, DLI only</td>
</tr>
<tr>
<td>STCLOAD</td>
<td>DS H</td>
<td>Physical loads</td>
</tr>
<tr>
<td>STCXTI</td>
<td>DS F</td>
<td>Time of excep's (10**-6 S)</td>
</tr>
<tr>
<td>STCPLOTI</td>
<td>DS F</td>
<td>Load time (10**-6 S)</td>
</tr>
<tr>
<td>STCZWAIT</td>
<td>DS F</td>
<td>ZTTA wait time (10**-3 S)</td>
</tr>
<tr>
<td>STCSWAIT</td>
<td>DS F</td>
<td>SGOC wait time (10**-3 S)</td>
</tr>
<tr>
<td>STCMILN</td>
<td>DS F</td>
<td>Length of input message (bytes)</td>
</tr>
<tr>
<td>STCMOLN</td>
<td>DS F</td>
<td>Length of output message (bytes)</td>
</tr>
<tr>
<td>STCPOLN</td>
<td>DS F</td>
<td>Length printer output mess. (bytes)</td>
</tr>
<tr>
<td>STCROLAL</td>
<td>DS F</td>
<td>Max. allocated roll area (bytes)</td>
</tr>
<tr>
<td>STCRULEN</td>
<td>DS F</td>
<td>Max. used roll area (bytes)</td>
</tr>
<tr>
<td>STCRCLEN</td>
<td>DS F</td>
<td>Max. used roll area after compress.</td>
</tr>
<tr>
<td>STCFRLEN</td>
<td>DS F</td>
<td>Max. size of loaded programs</td>
</tr>
</tbody>
</table>

*........DB CALL STATISTICS FOR TRACE.........................*

<table>
<thead>
<tr>
<th>Data Description</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STCDB1</td>
<td>DS F</td>
<td>Appendage 1:</td>
</tr>
<tr>
<td>STCDB1D</td>
<td>DS F</td>
<td>Direct access</td>
</tr>
<tr>
<td>STCDB1S</td>
<td>DS F</td>
<td>Sequential access</td>
</tr>
</tbody>
</table>

* STCDB2           | DS F      | Appendage 2: |
| STCDB2D          | DS F      | Direct access |
| STCDB2S          | DS F      | Sequential access |
*SAP ACCOUNTING Interface SAPSTEC (as of Release 4.3J, 4.4C and 5.0A)*

*STCDB3 DS F*  
APPENDAGE 3:  
STCDB3D DS F  
STCDB3S DS F  
DIRECT ACCESS  
SEQUENTIAL ACCESS

*STCDB4 DS F*  
APPENDAGE 4:  
STCDB4D DS F  
STCDB4S DS F  
DIRECT ACCESS  
SEQUENTIAL ACCESS

*STCDB5 DS F*  
APPENDAGE 5:  
STCDB5D DS F  
STCDB5S DS F  
DIRECT ACCESS  
SEQUENTIAL ACCESS

*........QUEUE NAME IN DOUT..................................*  
ORG STCPRLN  
STCDTQNA DS CL64

*........TIMER FOR ALL SAP DB CALLS (8*10**-6 S)..............*  
STCRUT DS F  
READ/UPDATE TIMER  
STCROT DS F  
READ/ONLY TIMER  
STCST DS F  
SET TIMER  
STCGT DS F  
GET TIMER  
STCWRT DS F  
REWRITE TIMER  
STCINT DS F  
INSERT TIMER  
STCDET DS F  
DELETE TIMER  
STCCALT DS F  
OTHER CALLS

*........TASK RELATED INFORMATION............................*  
STCWWAIT DS F  
WAIT ON SYNCHRONOUS UPDATE (MS)  
STCTWAIT DS F  
CPIC: WAIT ON TERMINAL INPUT (MS)  
STCDNR DS F  
DYNPRO NO. OF DIALOG STEP  
STCETIM DS F  
CREATION TIME OF APLZ RECORD  
STCROLL DS F  
ROLL KEY  
STCRTABS DS F  
RTAB CALLS SEQUENTIAL  
STCRTABD DS F  
RTAB CALLS DIRECT  
STCNRTC DS F  
NO. OF RTC/RTN’S  
STCLRTC DS F  
LONGEST INTERVAL BETWEEN RTC’S (MS)  
STCIMOD DS F  
# CREATED INTERNAL MODI  
STCTCPU DS XL8  
TIME STAMP CPU MEASUREMENT  
STCEND EQU *  
STCLENG EQU *-STCBEG

**Fields**

The fields have the following meaning and value ranges for SAP-ACCOUNTING-EXIT:

**STCTASK**

‘QM’ identifies the ACCOUNTING records written by the SAP transfer program

**STCFLAG1:**

not significant
STCMANDT
not significant

STCDATE
Date on which the transfer program was started (packed date format 'DDMMYY')

STCTIME
Time at which the transfer program was ended (packed time format 'HHMMSS')

STCTIMTR
Time at which the transfer program was started (Release 4.3: packed time format 'HHMMSS'
Release 5.0: ms after midnight)

STCRESP
Period for which the transfer program was activated (time between the start time and end time of
the transfer program in ms)

STCCTIM
not significant

STCATIM
not significant

STCTCODE
not significant

STCREPID
not significant

STCLTERM
Conversation identifier that the transfer program uses for identifying the LU6.2 connection to the
target system

STCACCT
not significant

STCREADU
No. of ‘read for update’ operations

STCREAD
No. of ‘read’ operations

STCSET
not significant

STCGET
not significant

STCWRITE
No. of ‘rewrite’ operations

STCINSRT
SAP ACCOUNTING Interface SAPSTEC (as of Release 4.3J, 4.4C and 5.0A)

No. of ‘insert’ operations

STCDELET

No. of ‘delete’ operations

STCLOAD

No. of ‘PGM load’ operations

STCSYNC

No. of ‘syncpoint’ operations

STCROLLB

No. of ‘rollback’ operations

STCCALL

No. of ‘other calls’

STCMOLN

Total no. of bytes transferred by the SAP transfer program (sum of bytes from the transferred SAP protocol information and the user data)

STCDTQNA

Name of the transferred DOUT queue

STCTWAIT

Total wait time of the transfer program in ms (consists of the wait time for the connection set up and the wait time when receiving transfer buffers)
Queue Interface for C Programs: RFC to R/3

**Definition**

You can use the SAP interface *Remote Function Call* (RFC) to call the function modules of the queue interface in R/3 from external C programs to place data in - or read data from - a queue. The necessary functions are provided by the RFC library.

**Use**

For details of the SAP interface RFC, see the documentation [Remote Communications [Extern]].

For details of the Q-API function modules, see [Queue Interface in the R/3 System][Seite 139].

An ABAP program calls a queue function module in another R/3 System (C11) as follows:

```plaintext
CALL FUNCTION 'QUEUE_OPEN'
  DESTINATION 'C11'
  EXPORTING
    NAME     = 'TESTQUEUE'
    OPENMODE = 'W'
    TYPE     = 'A'.
```

Accordingly, a C program calls the same queue function module in the R/3 System as follows:

```c
/**
 */

#include "saprfc.h"

char               queue[20] = 'TESTQUEUE';
char               openmode  = 'W';
char               qtype     = 'A';
int                rc;
char               *exception ;
RFC_OPTIONS       rfc_options;
RFC_PARAMETER     exporting[5],importing[5];
RFC_HANDLE        handle    = RFC_HANDLE_NULL;

: rfc_options.destination = "C11";
  handle = RfcOpen( &rfc_options); /* link to RFC Partner */

: exporting[0].name = "NAME";
  exporting[0].nlen = 4;
  exporting[0].type = TYPC;
  exporting[0].addr = queue;
  exporting[0].leng = 20;
```
Queue Interface for C Programs: RFC to R/3

```c
exporting[1].name = "OPENMODE";
exporting[1].nlen = 8;
exporting[1].type = TYPC;
exporting[1].addr = &openmode;
exporting[1].leng = 1;

exporting[2].name = "TYPE";
exporting[2].nlen = 4;
exporting[2].type = TYPC;
exporting[2].addr = &qtype;
exporting[2].leng = 1;

exporting[3].name = (char *)0; /* no more export parameters */

importing[0].name = NULL;      /* no import parameter(s) */

tables[0].name    = NULL;      /* no internal table(s) */

exception         = NULL;      /* default exception handling*/

rc = RfcCallReceive( handle,
                      "QUEUE_OPEN",
                      exporting,
                      importing,
                      tables,
                      &exception);
```

An ABAP/4 program calls a queue function module in another R/3 System (C11) as follows:

```
CALL FUNCTION 'QUEUE_OPEN'
  DESTINATION 'C11'
  EXPORTING
    NAME     = 'TESTQUEUE'
    OPENMODE = 'W'
    TYPE     = 'A'.
```

Accordingly, a C program calls the same queue function module in the R/3 System as follows:

```c
/***
 ***
 */

#include "saprfc.h"

char               queue[20] = 'TESTQUEUE';
char               openmode  = 'W';
char               qtype     = 'A';
int                rc;
char              *exception ;
RFC_OPTIONS        rfc_options;
RFC_PARAMETER      exporting[5],importing[5];
```
RFC_HANDLE    handle = RFC_HANDLE_NULL;

:  rfc_options.destination = "C11";
handle = RfcOpen( &rfc_options);   /* link to RFC Partner */
:
exporting[0].name = "NAME";
exporting[0].nlen = 4;
exporting[0].type = TYPC;
exporting[0].addr = queue;
exporting[0].leng = 20;
exporting[1].name = "OPENMODE";
exporting[1].nlen = 8;
exporting[1].type = TYPC;
exporting[1].addr = &openmode;
exporting[1].leng = 1;
exporting[2].name = "TYPE";
exporting[2].nlen = 4;
exporting[2].type = TYPC;
exporting[2].addr = &qtype;
exporting[2].leng = 1;
exporting[3].name = (char *)0; /* no more export parameters */
importing[0].name = NULL;      /* no import parameter(s) */
tables[0].name    = NULL;      /* no internal table(s)      */
exception         = NULL;      /* default exception handling*/
rc = RfcCallReceive( handle,
   "QUEUE_OPEN",
   exporting,
   importing,
   tables,
   &exception);
...

⚠️

Data objects which are read from a queue by an ABAP program must not exceed the field length of field APQD-VARDATA, which is defined in the Data Dictionary.
Using SAP Test Programs

In the following topics, you will learn how to test your CPI-C communication with the SAP test programs.

Available SAP Test Programs [Seite 175]

Specifying Program Parameters [Seite 177]

Requirements for Starting an External Partner Program [Seite 178]

Testing Connections [Seite 179]
Available SAP Test Programs

Definition
The following test programs are available:

Test programs

<table>
<thead>
<tr>
<th>Programs</th>
<th>Programs Written in C</th>
<th>ABAP programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling</td>
<td>ccpict1</td>
<td>acpict1</td>
</tr>
<tr>
<td>Callable</td>
<td>ccpict2</td>
<td>acpict2</td>
</tr>
</tbody>
</table>

Use
The test programs undertake CPI-C communication.

Each calling program (C or ABAP) can call one of the callable programs (C or ABAP).

These test programs therefore let you check program-to-program communication for all constellations.

ABAP test programs in R/2:
The SAP test programs are delivered with the CPI-C development libraries. The ABAP programs are directly available in new Releases of R/2 Systems. Otherwise you must upload these programs in your R/2 System.

Structure
There are some special features for ABAP programs and programs written in C.

Programs Written in C
The names of the C programs end with t or s. These letters specify which protocol the communication is based on.

<table>
<thead>
<tr>
<th>ccpictls</th>
<th>SNA is the protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccpictlt</td>
<td>TCP/IP is the protocol</td>
</tr>
</tbody>
</table>

You must create a side info file in your work directory before starting the calling C program (see section “Side Information Tables” in the documentation BC - SAP Communication: Configuration [Extern]).

The calling C program records the program activities on the screen (stdout). The C program called creates the trace file cpict2t.trc.

You can activate the CPI-C trace function before the start of the calling C program as follows:

- CPIC_TRACE=2 in the side info file
- Environment variable CPIC_TRACE=2
Available SAP Test Programs

Program-to-program communication normally takes place via the SAP gateway. You do not need the SAP gateway if the platform of each partner program supports the SNA protocol LU6.2. This is possible with the following constellations:

- C program ↔ R/2 program
- C program ↔ C program

Here, use the SNA-specific C programs.

**ABAP programs**

Both the calling and callable ABAP programs can be found in an R/2 or R/3 System.
The calling ABAP program records the program activities on the screen.
Specifying Program Parameters

Explanation You must specify parameters for the calling program. The calling program can be an ABAP or C program.

**ABAP program**

When executing the ABAP program, you must specify the following program parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEST</td>
<td>destination according to TXCOM/XCOM</td>
</tr>
<tr>
<td>ABAP</td>
<td>'&lt;'/X&gt; X, if the partner is acpict2</td>
</tr>
<tr>
<td>CONVERT</td>
<td>'&lt;'/X&gt; X, if conversion is required</td>
</tr>
<tr>
<td>USER</td>
<td>&lt;user&gt; SAP user name</td>
</tr>
<tr>
<td>PWD</td>
<td>&lt;Password&gt; password</td>
</tr>
</tbody>
</table>

*Testing a Connection [Seite 179]* only details those program parameters which require an entry.

**Program Written in C**

When executing a program written in C, you must specify the following program parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest</td>
<td>&lt;destination&gt; destination according to sideinfo</td>
</tr>
<tr>
<td>abap</td>
<td>if the partner is acpict2</td>
</tr>
<tr>
<td>conv</td>
<td>if conversion is required</td>
</tr>
<tr>
<td>usr</td>
<td>&lt;user&gt; SAP user name</td>
</tr>
<tr>
<td>pwd</td>
<td>&lt;Password&gt; password</td>
</tr>
</tbody>
</table>
Requirements for Starting an External Partner Program

There may be several reasons why a called CPI-C program cannot be started. First decide what type of program it is:

- A local program
- A remote program started via Remote Shell

Starting a Local Target Program

The SAP Gateway starts a local CPI-C program via fork/exec.

To avoid errors, ensure the following two conditions are met:

- The program is located in the search path of the Gateway ID.
- It is executable for the Gateway ID.

Log on with the Gateway ID and check whether the program to be started is in the search path of the Gateway ID:

**UNIX:** `which <program>`

Starting the Target Program via Remote Shell

A program is started on a remote computer via Remote Shell. For this, the following requirements must be met on the remote computer:

- The Gateway ID must exist.
- The Gateway compter must be entered in the .rhosts file. The .rhosts file must be located in the HOME directory or in the path of the Gateway ID.
- The program to be started must be installed in the HOME directory of the Gateway ID.

Log on with the Gateway ID and check, using Remote Shell, whether the authorizations necessary for calling a remote program exist, and that this is in the search path of the Gateway ID:

**UNIX:** `remsh <host> date
which <program>`

The Gateway processes in BS2000 are used exclusively for switching connections to R/2 Systems.

You cannot, for example, start programs outside the R/2 System in the BS2000 host via the SAP Gateway.
Testing Connections

When testing connections, you must specify various parameters, depending on whether the calling program is an ABAP/4 program or a C program:

You will find more information in the following topics:

- Calling Program: ABAP Program in R/3 [Seite 180]
- Calling Program: ABAP Program in R/2 [Seite 181]
- Calling Program: C Program [Seite 183]
Calling Program: ABAP Program in R/3

You call the test program `acpict1` with transaction SE38.

**Partner: ABAP Program in R/3**

Define the following program parameters:

```
DEST  <dest>  ABAP  X USER  <user name>  PWD  <Password>
```

You must define the following parameters in the TXCOM side info table:

```
DEST  LU  TP  Prot  <dest>  <R/3 host>
      <Dispatcher service>  I
```

**Partner: ABAP Program in R/2**

Define the following program parameters:

```
DEST  <destination>  ABAP  X CONVERT  X USER  <user name>  PWD  <Password>
```

In the TXCOM side info table, you only have to define the DEST and Prot parameters (LU and TP are ignored):

```
DEST  LU  TP  Prot  <dest>  -  -  C
```

You must define the following parameters in the side info table on the gateway platform:

```
DEST  =<dest>
LU    =<Logical unit>
TP    =X1SA
```

**Partner: Program Written in C**

Define the following program parameter:

```
DEST  <destination according to TXCOM>
```

You must define the following parameters in the TXCOM table:

```
DEST  LU  TP  Prot  <dest>  <partner host>
      ccppict2t  E
```
Calling Program: ABAP Program in R/2

You call the acpict1 test program with transaction TM38.

Partner: ABAP Program in R/3

Define the following program parameters:

```
<table>
<thead>
<tr>
<th>DEST</th>
<th>&lt;destination&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>X</td>
</tr>
<tr>
<td>CONVERT</td>
<td>X</td>
</tr>
<tr>
<td>USER</td>
<td>&lt;user name&gt;</td>
</tr>
<tr>
<td>PWD</td>
<td>&lt;Password&gt;</td>
</tr>
</tbody>
</table>
```

Here, the SAP communications programs `gwhost` for CICS or `gwims` for IMS are required for communication. Details on this can be found under “Parameters on the SNA Subsystem” in the documentation BC SAP Communication: Configuration [Extern].

In a BS2000 host, `gwhost` is required (Job SAPGWHO). For more information, see the section “Connection Setup by the R/2 System” in the documentation BC SAP Communication: Configuration [Extern].

You must define the following parameters in the XCOM side info table:

```
<table>
<thead>
<tr>
<th>DEST</th>
<th>LU</th>
<th>TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;dest&gt;</td>
<td>&lt;LU&gt;</td>
<td>&lt;Alias for gwhost or gwims&gt;</td>
</tr>
</tbody>
</table>
```

You must define the following parameters in the side info table on the gateway platform:

```
| DEST | =<Alias for gwhost or gwims> |
| GWHOST | =<SAP gateway host> |
| GWSERV | =<SAP gateway service> |
| PROTOCOL | =I |
| LU | =<R/3 host> |
| TP | =<SAP dispatcher service> |
```

Partner: ABAP Program in R/2

Define the following program parameters:

```
<table>
<thead>
<tr>
<th>DEST</th>
<th>&lt;destination&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>X</td>
</tr>
<tr>
<td>CONVERT</td>
<td>X</td>
</tr>
<tr>
<td>USER</td>
<td>&lt;user name&gt;</td>
</tr>
<tr>
<td>PWD</td>
<td>&lt;Password&gt;</td>
</tr>
</tbody>
</table>
```

You must define the following parameters in the XCOM side info table:

```
<table>
<thead>
<tr>
<th>DEST</th>
<th>LU</th>
<th>TP</th>
</tr>
</thead>
</table>
```
CALLING PROGRAM: ABAP PROGRAM IN R/2

<dest> <LU> X1SA

- Communication between R/2 Systems is only possible on IBM hosts when CICS is used as the data communications system.
- Local communication on an R/2 IBM host is not possible. CICS does not support a local Conversation via SNA-LU6.2.
- A local Conversation is possible for SNI UTM systems.

PARTNER: PROGRAM WRITTEN IN C

As the partner is not an SAP program, you only have to specify the following program parameter:

DEST <destination>

Here, the SAP communications programs gwhost for CICS or gwims for IMS are required for communication. Details on this can be found under "Parameters on the SNA Subsystem Platform with R/2" in the documentation BC - SAP-Communication: Configuration [Extern].

In a BS2000 host, gwhost is required (Job SAPGWHO). For more information, see the section “Connection Setup by the R/2 System” in the documentation BC - SAP Communication: Configuration [Extern].

You must define the following parameters in the XCOM side info table:

DEST LU TP
<dest> <LU> <Alias for gwhost or gwims>

You must define the following parameters in the side info table on the gateway platform:

DEST=<Alias for gwhost or gwims>
GHOST=<SAP gateway host>
GWSERV=<SAP gateway service>
PROTOCOL=E
LU=<partner host>
TP=ccpict2t
Calling Program: Program Written in C

Call the ccpict1t C program with the appropriate parameters.

Partner: ABAP Program in R/3

You must define the following parameters in the sideinfo file for the calling program:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEST</td>
<td>&lt;user defined destination&gt;</td>
</tr>
<tr>
<td>LU</td>
<td>&lt;R/3 host, on which ccpict2 is running&gt;</td>
</tr>
<tr>
<td>TP</td>
<td>&lt;SAP dispatcher service&gt;</td>
</tr>
<tr>
<td>GWHOST</td>
<td>&lt;SAP gateway host&gt;</td>
</tr>
<tr>
<td>GWSERV</td>
<td>&lt;SAP gateway service&gt;</td>
</tr>
<tr>
<td>PROTOCOL</td>
<td>I</td>
</tr>
</tbody>
</table>

You call the ccpict1t test program as follows:

ccpict1t -dest <DEST according to sideinfo> -abap -usr <SAP user name> -pwd <password>

Partner: ABAP Program in R/2

You must define the following parameters in the sideinfo file for the calling program:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEST</td>
<td>&lt;user defined destination&gt;</td>
</tr>
<tr>
<td>GWHOST</td>
<td>&lt;SAP gateway host&gt;</td>
</tr>
<tr>
<td>GWSERV</td>
<td>&lt;SAP gateway service&gt;</td>
</tr>
<tr>
<td>PROTOCOL</td>
<td>C</td>
</tr>
</tbody>
</table>

You must define the following parameters in the sideinfo file for the SAP gateway:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEST</td>
<td>&lt;user defined destination&gt;</td>
</tr>
<tr>
<td>LU</td>
<td>&lt;Logical unit&gt;</td>
</tr>
<tr>
<td>TP</td>
<td>X1SA</td>
</tr>
<tr>
<td>PROTOCOL</td>
<td>C</td>
</tr>
</tbody>
</table>

Note the platform-specific special features of side info entries (see BC - SAP Communication: Configuration [Extern]).

You call the ccpict1t test program as follows:

ccpict1t -dest <DEST in sideinfo> -abap -conv -usr <SAP user name> -pwd <password>

Partner: Program Written in C

You must define the following parameters in the sideinfo file for the calling program:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEST</td>
<td>&lt;user defined destination&gt;</td>
</tr>
<tr>
<td>LU</td>
<td>&lt;host, on which ccpict2 is to run&gt;</td>
</tr>
</tbody>
</table>
Calling Program: Program Written in C

<table>
<thead>
<tr>
<th>TP</th>
<th>= ccpict2</th>
</tr>
</thead>
<tbody>
<tr>
<td>GWHOST</td>
<td>=&lt;SAP gateway host&gt;</td>
</tr>
<tr>
<td>GWSERV</td>
<td>=&lt;SAP gateway service&gt;</td>
</tr>
<tr>
<td>PROTOCOL</td>
<td>=E</td>
</tr>
</tbody>
</table>

You call the ccpict1t test program as follows:

ccpict1t -dest <user defined destination>
Error Analysis

Error analysis is described in the following topics:

Function SAP_CMPERR [Seite 186]

Error Analysis Under OS/2 [Seite 187]

Error Analysis Under UNIX and WindowsNT [Seite 189]

Error Messages of the SAP Transfer Program [Seite 191]
Function **SAP_CMPERR**

**Use**

If you use the function SAP_CMPERR in the SAP development library for your C program, you get a short error description if an error occurs.

```c
CM_RETCODE return_code;
..;
CMALLC(convid,&return_code);
if (return_code != CM_OK)
{
    printf("SAP-INFO: %s\n", SAP_CMPERR());
}
..;
```
Error Analysis Under OS/2

Purpose

CPI-C Trace
You can run an SAP trace while programs communicate via the CPIC interfaces.
You can activate and deactivate the trace via the environment variable:

\[ \text{CPIC\_TRACE} = [1,2,3] \]

For each process that the SAPCPIC interface uses, a log file with the name CPxxxxx.TRC is created. xxxxx is the number of the process.

Error Log File
If an error occurs during communication, this error is logged in the file CPxxxxx.ERR with xxxxx = number of the process, in which the error occurred.

Error Messages
If an error occurs during communication, the error is logged. The error messages are output as described in the APPC/LU6.2 guide. In addition, SAP has defined a series of error messages, which are listed below.

These error messages are not CPI-C return codes, and cannot therefore be intercepted in the program.
They simply provide additional information in the trace or error log, which should facilitate error recovery.

\[ \text{SAP\_TABLE\_NOT\_FOUND:} \]
The file SAPCPIC.TBL was not found (CMINIT) or SAPCPIC.TBL cannot be accessed.

\[ \text{SAP\_INVALID\_DESTINATION:} \]
The specified symbolic destination was not found in SAPCPIC.TBL (CMINIT, CMACCP).

\[ \text{SAP\_TOO\_FEW\_ARGUMENTS:} \]
Not all fields in SAPCPIC.TBL are filled for a particular symbolic destination sind.

\[ \text{SAP\_NO\_FREE\_SIDE\_TABLE\_SLOT:} \]
An attempt was made to exceed the maximum number of simultaneously active SNA conversations.

\[ \text{SAP\_CANT\_GET\_LOCAL\_PGM\_NAME:} \]
The attempt to determine the name of the local program failed. The corresponding OS/2 return code is in the error log.

\[ \text{SAP\_INVALID\_CPIC\_CV\_ID:} \]
An invalid conversation ID was passed to a SAPCPIC interface function.
Error Analysis Under OS/2

SAP_CANT_GET_SIDE_INFO_SEM, SAP_CANT_LOCK_SIDE_INFO_SEM, SAP_CANT_RELEASE_SIDE_INFO_SEM:
SAP’s CPI-C interface works internally with system semaphores. These three errors indicate that either the creation, locking or release of a system semaphore failed. The interface cannot continue to work correctly.

The corresponding OS/2 return code is in the error log.

SAP_CANT_GET_PROCESS_PID:
The attempt to determine the number of a process failed. The corresponding OS/2 return code is in the error log.

SAP_CANT_GET_INFOSEG:
The attempt to get OS/2 system information failed. The corresponding OS/2 return code is in the error log.

SAP_TOO_MANY_CPIC_TBL_ENTRIES:
More connections are defined in SAPCPIC.TBL than the SAPCPIC interface can administer. The current maximum value = 100 connections.

SAP_CANT_GET_SIDE_INFO_SHM, SAP_CANT_GET_DISPLAY_INFO_SHM:
The SAPCPIC interface works internally with shared memory. These two errors indicate that the creation of shared memory areas failed. The interface cannot continue to work correctly. The corresponding OS/2 return code is in the error log.

SAP_NO_FREE_SESSION_AVAILABLE:
All SNA connections defined for a particular symbolic destination are in use, i.e., all SNA sessions have conversations.

SAP_INVALID_LU_NAME:
An invalid local SNA Logical Unit name was entered in SAPCPIC.TBL for a particular symbolic destination.

SAP_INVALID_PLU_NAME:
An invalid local SNA Partner Logical Unit name was entered in SAPCPIC.TBL for a particular symbolic destination.

SAP_INVALID_MODE_NAME:
An invalid SNA mode name name was entered in SAPCPIC.TBL for a particular symbolic destination.
Error Analysis Under UNIX and WindowsNT

Purpose

CPI-C Trace
If the function SAP_CMPERR is not sufficient for error analysis, you must activate the CPI-C trace. You have several options:

- Set the shell variable CPIC_TRACE
- Entry in the side information file
- Via the target program for R/2 host-to-workstation communication

These options are described below.

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No trace</td>
</tr>
<tr>
<td>1</td>
<td>Error trace</td>
</tr>
<tr>
<td>2</td>
<td>Trace flow and shortened data trace</td>
</tr>
<tr>
<td>3</td>
<td>Trace flow and complete data trace</td>
</tr>
</tbody>
</table>

The trace will be in a trace file (CPICTR<PID>) in the current work directory.

The trace function for the called program can only be activated via the shell variable CPIC_TRACE.

If you want to check a configuration file, you can use the CPI-C test programs cpi<it>ct1</it> and cpi<it>c</it>t2.

Activation via Environment Variable
The environment variable has priority over an entry in the side information file.

You can use the environment variable CPIC_TRACE to activate a trace function, as in the following:

UNIX: setenv CPIC_TRACE 3

Activation via the Side Information File
If, for example, you want to record an error trace, make the following entry in the side information file:

CPIC_TRACE=2

Activation via the Target Program
If you want to start an R/3 or C program on a remote workstation from an R/2 host, you normally set the trace level using the environment variable CPIC_TRACE. If this is not possible, (for example, if login or cshrc are not processed), define the trace level in the side information file using the variable CPIC_TRACE.

The trace level set in this way can be activated via the called program using the following function:

CMSetCPICTrace()
main()
{
    CMSetCPICTrace();
    SAP_CMACCP();
    CMACCP();
}
Error Messages of the SAP Transfer Program

Definition

Error messages are sent by the receiving program to the transfer program. The transfer program enters the messages in the corresponding fields of the queue administration information record.

In addition to these error messages, these fields also contain messages regarding local errors of the transfer program.

Structure

If local error situations occur with the transfer program, the following error messages are entered in the queue administration information record:

'AKCS PARTNER DOES NOT RESPOND

When establishing a connection (for example, EXEC CICS ALLOCATE), the wait for the connection is terminated after a certain wait time because the target system is not active (for example, the Communications Manager is not started on the workstation).

'ATCV INVALID FUNCTION FOR CURRENT LU6.2 STATE

The LU6.2 function called by the transfer program cannot be executed with the current status of the connection to the target system (for example, the target application is not adhering to the send and receive sequence defined by the transfer data type).

'AISS LU6.2 SECURITY ERROR

The target application, to which the transfer program is to build a connection, is performing an authorization check, which is not met by the transfer program (for example, in the Communication Manager on the workstation, conversation security is active for the started transaction program (remotely attachable transaction program')).

'E001 CANNOT ALLOCATE LU6.2 SESSION

The attempt to establish a connection is rejected by the network (for example, the specified target system is not known).

'E002 CANNOT CONNECT PROCESS

The connection to the transaction program in the target system is rejected (for example, the specified transaction program is not defined as a startable transaction program).

'E003 CANNOT SEND BUFFER WITH INVITE

Error sending data to the target transaction program.

'E004 CANNOT RECEIVE BUFFER

Error receiving data from the target transaction program (for example, the specified transaction program is not defined as a startable transaction program).

'E005 CANNOT FREE LU6.2 SESSION

Error establishing a connection with the target transaction program.

'E006 CANNOT SEND BUFFER

Error sending data to the target transaction program.
Error Messages of the SAP Transfer Program

'E007 FREE SESSION INDICATOR RECEIVED
The target transaction has already established the connection although the transfer program has not yet completed processing. The target application is not adhering to the send and receive sequence defined by the transfer data type or an error situation has occurred, in which it can no longer communicate the error with an error message buffer.

'E008 CANNOT SEND BUFFER WITH LAST WAIT
Error sending data to the target transaction program.

'E009 CONVERSATION PARTNER DO NOT INVITE TO SEND
The target transaction is not returning the send authorization to the transfer program. The target application is not adhering to the send and receive sequence defined by the transfer data type.

'E014 EIBERRCD RECEIVED
The transfer program has received an error message from the LU6.2 function that was performed. The EIBERRCD error is displayed as an 8-digit number after the string 'E014'. The cause of the error is described in the CICS/OS/VS Intercommunications Facilities Guide.

'E021 CANNOT RECEIVE RETURN BUFFER
Error receiving data from the target transaction program.

'E025 ACCESSING DOUT TO GET DATA TO BE SEND
Internal error accessing the queue to update send data or status information.

'E034 ACCESSING DOUT TO SET PROCESS INACTIVE INDICATOR'
Internal error during access to update status information.

'E039 CANNOT UPDATE APQI/APQD RECORD
Internal error accessing the queue to update send data or status information.

'E040 CANNOT DELETE APQD RECORDS
Internal error accessing the queue to update send data.

'E304 CANNOT UNLOCK/ENDBR DOUT AFTER RETRIEVING PARAM
Internal error during access to update status information.

'E305 WARNING NO INPUT PARAMETER RECEIVED
The transfer program was not started by a 'EXEC CICS START' command.

'E306 CANNOT RETRIEVE COMMAREA
The transfer program was started interactively instead of by a 'EXEC CICS START' command. The name of the queue to be started was not received.

'E310 ACCESSING DOUT TO SET PROCESS ACTIVE INDICATOR
Internal error during access to update status information.

'E315 RETRIEVED KEY FIELD IS TOO SHORT
The queue name passed during startup does not have the correct length.
'**E320 NOT SUPPORTED DRIVER TYPE**'  
The transfer program was started for a transfer data type, which it does not support.

'**E325 ACCEPT ONLY RECEIVER CORRECTION RESPONSIBILITY**'  
The transfer program can only run if the target transaction program performs the error correction.

'**E330 OUTPUT PROCESS IS ALREADY ACTIVE**'  
A transfer program is already active for this queue.

'**E340 NO RECORD FOUND TO SEND TO THE TARGET SYSTEM**'  
The transfer program was started for the queue, but has not found any transfer data to be sent in the queue.

'**E530 START BROWSE SPLD DATASET**'  
Internal error accessing the queue to update send data to be read from the SAP spool.

'**E550 NO SPOOL CONTROL RECORD FOUND**'  
Internal error accessing the queue to update send data to be read from the SAP spool.

'**E555 WRONG SPOOL FILE PASSWORD**'  
The SAP spool file, from which the transfer data was to be read, is password protected. The SAP spool password in the queue entry does not match the password of the referenced SAP spool file.

'**E560 SPOOL RECORD WITH ZERO FIELD LENGTH**'  
Internal error accessing the queue to update send data to be read from the SAP spool.

'**UNDEFINED ERROR CODE IN DRIVER TASK**'  
Internal error

In addition to the error text, an error code is output with the error message. As with the error message texts, there is a difference between the local error codes and the error codes sent by the partner transaction program. Below is a list of local error message codes of the SAP transfer program in a CICS environment. The error message codes represent the CICS condition codes, which can occur during error recognition by the SAP transfer program. The local error message codes are listed opposite the corresponding CICS condition codes.

<table>
<thead>
<tr>
<th>Local error code</th>
<th>CICS condition code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBID</td>
<td>CBIDERR</td>
</tr>
<tr>
<td>DSDI</td>
<td>DISABLED</td>
</tr>
<tr>
<td>DSID</td>
<td>DSIDERR</td>
</tr>
<tr>
<td>DUPK</td>
<td>DUPKEY</td>
</tr>
<tr>
<td>DUPR</td>
<td>DUPREC</td>
</tr>
<tr>
<td>ENDF</td>
<td>ENDFILE</td>
</tr>
<tr>
<td>ILLO</td>
<td>ILLOGIC</td>
</tr>
<tr>
<td>INVR</td>
<td>INVREQ</td>
</tr>
<tr>
<td>IO</td>
<td>IOERR</td>
</tr>
<tr>
<td>ISCI</td>
<td>ISCINVREQ</td>
</tr>
<tr>
<td>ITEM</td>
<td>ITEMERR</td>
</tr>
<tr>
<td>LENG</td>
<td>LENGERR</td>
</tr>
</tbody>
</table>
## Error Messages of the SAP Transfer Program

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOST</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOSTG</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOSP</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOSPACE</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOAL</td>
<td>Not Allocated</td>
</tr>
<tr>
<td>NOTALLOC</td>
<td>Not Allocated</td>
</tr>
<tr>
<td>NOAU</td>
<td>Not Authorized</td>
</tr>
<tr>
<td>NOTAUTH</td>
<td>Not Authorized</td>
</tr>
<tr>
<td>NOTF</td>
<td>Not Found</td>
</tr>
<tr>
<td>NOTFND</td>
<td>Not Found</td>
</tr>
<tr>
<td>NOTO</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOTOPEN</td>
<td>Not Opened</td>
</tr>
<tr>
<td>QID</td>
<td>Queue ID Error</td>
</tr>
<tr>
<td>QIDERR</td>
<td>Queue ID Error</td>
</tr>
<tr>
<td>SESS</td>
<td>Session Busy</td>
</tr>
<tr>
<td>SESSBUSY</td>
<td>Session Busy</td>
</tr>
<tr>
<td>SIGN</td>
<td>Signal</td>
</tr>
<tr>
<td>SIGNAL</td>
<td>Signal</td>
</tr>
<tr>
<td>SYID</td>
<td>System ID Error</td>
</tr>
<tr>
<td>SYSIDERR</td>
<td>System ID Error</td>
</tr>
<tr>
<td>TERM</td>
<td>Term Error</td>
</tr>
<tr>
<td>TERMERR</td>
<td>Term Error</td>
</tr>
<tr>
<td>DSID</td>
<td>Disabled</td>
</tr>
<tr>
<td>DISABLED</td>
<td>Disabled</td>
</tr>
<tr>
<td>DUPK</td>
<td>Duplicate Key</td>
</tr>
<tr>
<td>DUPKEY</td>
<td>Duplicate Key</td>
</tr>
<tr>
<td>NOSP</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOSPACE</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOTA</td>
<td>Not Authorized</td>
</tr>
<tr>
<td>NOTAUTH</td>
<td>Not Authorized</td>
</tr>
<tr>
<td>NOTF</td>
<td>Not Found</td>
</tr>
<tr>
<td>NOTFND</td>
<td>Not Found</td>
</tr>
<tr>
<td>NOTO</td>
<td>Not Opened</td>
</tr>
<tr>
<td>NOTOPEN</td>
<td>Not Opened</td>
</tr>
</tbody>
</table>
Special Features on R/2 Hosts

Special features on various R/2 hosts are explained in the following chapters:

BS2000 R/2 Host: UTM-UTM Connection [Seite 196]
MVS/VSE R/2 Host: CICS and IMS [Seite 201]
BS2000 R/2 Host: UTM-UTM Connection

Purpose

No CPI-C interface is available in BS2000. Alternatively, you can use program-to-program communication between ABAP programs and UTM subprograms based on UTM-VTV. This means that two R/2 Systems under UTM are connected.

Implementation considerations

Communication can be initiated by both sides. As UTM does not support calls equivalent to the CPI-C calls Initialize, Allocate, Accept and Deallocate, these calls need not be used or must be simulated as follows:

<table>
<thead>
<tr>
<th>CPI-C call in ABAP</th>
<th>UTM call</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMUNICATION ALLOCATE</td>
<td>APRO</td>
</tr>
<tr>
<td>COMMUNICATION SEND</td>
<td>MPUT</td>
</tr>
<tr>
<td>COMMUNICATION RECEIVE</td>
<td>MGET</td>
</tr>
</tbody>
</table>

To create UTM-D subprograms, you will require the corresponding UTM guides. The rules for distributed transaction processing must be adhered to.

The following examples are discussed:

Initiator: R/2 System [Seite 199]
Initiator: Non-SAP System [Seite 200]

Features

- UTM-D allows communication within the same application.
  
  Both the sending and the receiving program can be run for test purposes in the same SAP system. For the test, you do not need a second application (see the generation notes).

- Asynchronous CPI-C communication
  
  A special feature contained only in UTM implementation is the possibility of asynchronous communication. Unlike synchronous communication, which always involves sending a message (CMSEND) and then waiting for a reply (CMRECEIVE), asynchronous communication only involves sending.

  This offers a series of advantages:
  
  - When sending, the partner application need not be active.
  - It is possible to send from asynchronous UTM transactions, i.e. from stations not operated.
  - Easier programming
  - No problems with UTM error handling

  Up to now, the asynchronous procedure has been used predominantly for sending messages to the SAP system. The procedure is based on a special feature of UTM-D,
that is asynchronous partial messages from the sender can be presented to the receiver as individual messages. Asynchronous communication from the external system to the SAP system is operated as follows:

**External System**

```
INIT
FGET / MGET
APRO AM Ipap, Itac
FPUT 'CONNCPI(C1..');
FPUT first data record
FPUT second data
FPUT third data record
FPUT n-th data record
PEND RE/FI
```

**SAP System**

```
ACCEPT
RECEIVE first data
RECEIVE second data
RECEIVE third data
RECEIVE n-th data
```

You send a logon message (in which you specify the receiving report), and can then send as many data messages as required up to the maximum number of partial messages possible (around 32000 Bytes). The requirements here are simply large enough page pools in the KDCFILEs of sender and receiver.

**Constraints**

- No debugging mode in the ABAP program
  
  ABAP programs, which contain the COMMUNICATION statements, cannot run in debugging mode.

  Possible intermediate dialogs in this mode violate the rules for distributed transaction processing with UTM-D. Violation results in a PEND ER and task termination.

- No character string with FREE
  
  Free communication is restricted in that no message can begin with the string FREE because this is used for the DEALLOCATE simulation.

- Normally, the initiator (PLU) of communication must also end communication.

As the statement DEALLOCATE is not supported in UTM, it must be simulated. From the point of view of the SAP system on the host, there are two synchronous processing scenarios:

1. DEALLOCATE statement in the PLU
   (Primary Logical Unit)

   If the PLU performs a DEALLOCATE statement, which follows a RECEIVE, this must be made known to the SLU. A FREE message is created in the service routine `Dealocate` and sent to the SLU. At the same time, terminal output from the PLU is delayed until the response required by the UTM protocol is received from the SLU.
2. DEALLOCATE statement in SLU  
(Secondary Logical Unit)

If, for example, an SLU encounters an error situation, in which it is preferable to  
continue communication, the SLU must end communication using DEALLOCATE.  

The SLU has two possible courses of action:

- Send a FREE message to the PLU  
  (possibly with an error message)
- Perform a DEALLOCATE call

If DEALLOCATE is called, the FREE message is generated automatically. You  
cannot send a message and then perform a DEALLOCATE call.

If the SLU is not an SAP System, a process termination by PEND FI of the SLU is  
interpreted as DEALLOCATE by the SAP System.

A PLU can only close itself (PEND FI) after all its SLUs are closed. Otherwise, this would cause a  
UTM task termination with 87Z and KS01 (see bottom-up strategy in the UTM documentation).
## Initiator: R/2 System

In the example below, the calling program is an ABAP program.

<table>
<thead>
<tr>
<th>ABAP program</th>
<th>UTM subprogram</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMUNICATION INIT ID convid</td>
<td>INIT MGET message from SAP</td>
</tr>
<tr>
<td>DESTINATION sym_dest.</td>
<td>MPUT message to SAP PEND RE</td>
</tr>
<tr>
<td>COMMUNICATION ALLOCATE ID convid</td>
<td>INIT MGET.Deallocate simulation</td>
</tr>
<tr>
<td>COMMUNICATION SEND ID convid</td>
<td>MPUT Deallocate acknowledgement</td>
</tr>
<tr>
<td>BUFFER sendarea</td>
<td>PEND F I</td>
</tr>
<tr>
<td>LENGTH sendlen</td>
<td></td>
</tr>
<tr>
<td>COMMUNICATION RECEIVE ID convid</td>
<td>INIT MGET.Deallocate</td>
</tr>
<tr>
<td>BUFFER rcvarea</td>
<td>simulation</td>
</tr>
<tr>
<td>LENGTH rcvlen</td>
<td>MPUT.Deallocate</td>
</tr>
<tr>
<td>RECEIVED rcvlen</td>
<td>acknowledgement</td>
</tr>
<tr>
<td>DATAINFO datai</td>
<td>PEND F I</td>
</tr>
<tr>
<td>STATUSINFO stati</td>
<td></td>
</tr>
<tr>
<td>COMMUNICATION DEALLOCATE ID convid</td>
<td></td>
</tr>
</tbody>
</table>
## Initiator: Non-SAP system

In the example below, the calling program is part of a non-SAP system.

<table>
<thead>
<tr>
<th>UTM subprogram</th>
<th>ABAP program</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT MGET Terminal message APRO SAP system MPUT to SAP system PEND RE/KP</td>
<td>COMMUNICATION ACCEPT ID convid. COMMUNICATION RECEIVE ID convid BUFFER rcvarea LENGTH reqlen RECEIVED rcvlen DATAINFO datai STATUSINFO stati. COMMUNICATION SEND ID convid BUFFER sendarea LENGTH sendlen.</td>
</tr>
<tr>
<td>INIT MGET message from SAP MPUT to SAP system (Dealocate simulation) PEND RE/KP</td>
<td>COMMUNICATION RECEIVE ID convid BUFFER rcvarea LENGTH reqlen RECEIVED rcvlen DATAINFO datai STATUSINFO stati.</td>
</tr>
<tr>
<td>INIT MGET Deallocate acknowledgement from SAP MPUT Terminal message PEND RE/FI</td>
<td></td>
</tr>
</tbody>
</table>
MVS/VSE R/2 Host: CICS and IMS

Purpose

Make note of the following special features in the DC systems CICS and IMS when using an MVS/VSE R/2 host:

CICS Special Features [Seite 202]
IMS Special Features [Seite 203]
CICS Special Features

ABAP supports the *CPI-C Starter Set*. When using the LU6.2 program interface for CICS, this means:

- Mapped Conversation
- SYNCLEVEL 0 (NONE)

The table below contains a list of the CPI-C functions and the corresponding calls of the LU6.2 program interface (API) in CICS.

### CPI-C calls and CICS calls

<table>
<thead>
<tr>
<th>CPI-C in ABAP</th>
<th>CICS LU6.2 API</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td></td>
</tr>
<tr>
<td>ALLOCATE</td>
<td>EXEC CICS ALLOCATE + EXEC CICS CONNECT</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>EXEC CICS RECEIVE</td>
</tr>
<tr>
<td>SEND</td>
<td>EXEC CICS SEND</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>EXEC CICS RECEIVE</td>
</tr>
<tr>
<td>DEALLOCATE</td>
<td>EXEC CICS SEND LAST + EXEC CICS FREE</td>
</tr>
</tbody>
</table>

You must always analyze the internal CICS EIB fields to check the return codes (for example, EIBERRCD). If errors occur, you must check the entries in the SAP system log.

If you transfer large data volumes, you should use frequent RECEIVE calls to ensure that data transfer has physically taken place and the control of the SAP Basis system is returned.
IMS Special Features

The individual parameters required for LU6.2 communication are listed in the description of the IBM LU6.1 adapter for LU6.2 applications.

SAP is subject to all the restrictions of the LU6.1 adapter for LU6.2 applications.

For communication with an R/2 System in IMS environment, the following parameters are particularly important:

Active connection:

- The active establishment of a connection from the SAP system is only possible to the partner program IMSASYNC in asynchronous mode.

  Here, asynchronous means that a RECEIVE statement is not permitted, but the conversation must be acknowledged using CONFIRMED (Status = CM_CONFIRM_DEALLOC_RECEIVED).

- An IMSASYNC program should always be defined to receive and acknowledge data to be processed asynchronously from the IMS Message Queue.

Remotely attachable ABAP program:

- The SAP service transaction xxxX1SA is conversational.

- Several RECEIVE calls in sequence are not permitted. There must be at least one SEND call between two RECEIVE calls. The program status is displayed in the status field.

- The length of a transferred message must always be larger than zero.

- If several SEND statements are sent in succession, or very long messages are sent, segmentation will occur in the IMS receiver system.

  Only the first segment of a message is processed (up to 5.0C), so that data can be cut off.

  From R/2 Release 5.0D, the individual segments are concatenated and passed to the SAP application as a complete message.

- An external DEALLOCATE or a termination of the partner program are not passed on to the SAP system. For this reason, DEALLOCATE must always be called in the ABAP program in the IMS system.

- External security systems are not supported directly.

  Alternatively, you can call the command /SIGN. Use SEND to send the USERID/PWD to IMS. Use RECEIVE to get the corresponding IMS message: DFSxxx....
Sample Programs

This section contains example programs for R/3 and the R/2 Releases 5.0 and 4.3/4.4, as well as the APC and APQ headers DSECT for assembler programs (R/2 version 4.3/4.4).

- Sample Program for R/3 [Seite 205]
- Sample Programs for R/2 Release 5.0 [Seite 208]
- Sample Programs for R/2 Release 4.3/4.4
- APC and APQ Headers DSECT for Assembler Programs [Seite 216]
Sample Program for R/3

PROGRAM RSQAPI10.
*-----------------------------------------------------------------------------*
* This queue test program writes data into a queue.                           *
*-----------------------------------------------------------------------------*
INCLUDE RSEBCASC. " ASCII/EBCDIC conversion table
INCLUDE RSQAPIDF. " QAPI Defines

PARAMETERS:

QUEUE LIKE APQI-QID DEFAULT 'Q123',
TYPE LIKE APQI-QATTRIB DEFAULT Q_APPENDABLE,
DESTSYS LIKE APQI-DESTSYS DEFAULT '        ',
DATATYPE LIKE APQI-DATATYP DEFAULT '    ',
CLIENT LIKE APQI-MANDANT DEFAULT '   ',
PROGRAM LIKE APQI-PROGID DEFAULT '        ',
FORM LIKE APQI-FORMID DEFAULT '    ',
DRIVER LIKE APQI-STARTPGID DEFAULT 'RSQAPI20',
START LIKE APQI-STARTMODE DEFAULT Q_USERSTART,
DATE LIKE APQI-STARTDATE DEFAULT '        ',
TIME LIKE APQI-STARTTIME DEFAULT '      ',
ASCTOEBC DEFAULT 'X',
TESTLOOP TYPE I DEFAULT '5'.

DATA:

COUNTER TYPE I,

BUFFER LIKE APQD-VARDATA,
LENGTH TYPE I,
UNIT TYPE I,
POS TYPE I,
STATE,
USERID LIKE APQI-USERID VALUE 'CPIC2',
PASSWD LIKE APQI-PASSWD VALUE 'TEST'.

*BREAK-POINT.

*-----------------------------------------------------------------------------*
* Open a Queue                                                               *
*-----------------------------------------------------------------------------*
CALL FUNCTION 'QUEUE_OPEN'  
EXPORTING  NAME        = QUEUE " Queue ID
TYPE        = TYPE " Appendable/Unique
OPENMODE    = Q_WRITE " Write Mode

transfer is used
DESTINATION = DESTSYS " TXCOM entries are
needed
if partner is an ABAP
program
DATATYPE    = DATATYPE
CLIENT      = CLIENT

April 2001 205
Sample Program for R/3

USERID = USERID
PASSWORD = PASSWD
PROGRAM = PROGRAM
FORM = FORM

TXCOM entry
* fields are needed if partner is a ABAP/4 program
DATATYPE = DATATYPE
CLIENT = CLIENT
USERID = USERID
PASSWORD = PASSWD
PROGRAM = PROGRAM
FORM = FORM

* queue processing options
DRIVER = DRIVER
START = START
DATE = DATE
TIME = TIME

* import automatic generated name, if input was space
IMPORTING NAME = QUEUE.

*-------------------------------------------------------------*
* Put Data into Queue                                         *
*-------------------------------------------------------------*
DO TESTLOOP TIMES.

* first msg of unit
STATE = Q_FIRST.
BUFFER = 'QTEST00020LINE 1 of Queue Test'.
LENGTH = STRLEN( BUFFER).
PERFORM QUEUE_PUT USING QUEUE STATE LENGTH BUFFER.

* middle msg of unit
STATE = Q_MIDDLE.
BUFFER = 'QTEST00020LINE 2 of Queue Test'.
LENGTH = STRLEN( BUFFER).
PERFORM QUEUE_PUT USING QUEUE STATE LENGTH BUFFER.

* last msg of unit
STATE = Q_LAST.
BUFFER = 'QTEST00020LINE 3 of Queue Test'.
LENGTH = STRLEN( BUFFER).
PERFORM QUEUE_PUT USING QUEUE STATE LENGTH BUFFER.

ENDDO.

*----------------------------------------------------------------*
* Close Queue                                                   *
*----------------------------------------------------------------*
CALL FUNCTION 'QUEUE_CLOSE'
    EXPORTING NAME = QUEUE
    OPENMODE = Q_WRITE.
COMMIT WORK.
FORM: QPUT
-----------
Put a message into specified queue

FORM QUEUE_PUT USING QUEUE STATE LENGTH BUFFER.

WRITE: /(72) BUFFER.

IF ASCTOEBC = 'X'.
   TRANSLATE BUFFER TO CODE PAGE '0100'. "alternatively
ENDIF.

CALL FUNCTION 'QUEUE_PUT'
   EXPORTING NAME   = QUEUE
                   STATE  = STATE
                   LENGTH = LENGTH
                   BUFFER = BUFFER.

   IF ( STATE = Q_LAST OR STATE = Q_SINGLE).
      COMMIT WORK.
   ENDIF.
ENDFORM.
Sample Programs for R/2 Release 5.0

The ABAP sample programs contain functions and parameters which are dependent on the transfer data type.

- Data transfer in RODC format
  
  Program RSAPPQ10 [Seite 209]:

- Data transfer from the SAP spool system in CPIC format
  
  Program RSAPPQ20 [Seite 212]:

- Data transfer in CPIC format
  
  Program RSAPPQ30 [Seite 214]:
Program RSAPPQ10

***********************************************************************
*                                                           *
* Sample program for asynchronous communication via the DOUT *
* file with transfer data in Remote ODC format                *
* (transfer format RODC)                                      *
***********************************************************************
PARAMETERS:
  QDEST(8)  TYPE C DEFAULT 'K44     ',
  QAPPL(8)  TYPE C DEFAULT 'X1SA    ',
  QUSER(12) TYPE C DEFAULT 'RODC-TEST   ',
  QPASS(8)  TYPE C DEFAULT 'TEST    ',
  QSTDA     TYPE D DEFAULT SY-DATUM,
  QSTTI     TYPE T DEFAULT SY-UZEIT,
  QSTART(1) TYPE C DEFAULT 'A',
  QANZHL(5) TYPE C DEFAULT '2'.
***********************************************************************
* QDEST : Connection name of target system                    *
* QAPPL : Transaction program name in target system           *
* (X1SA=SAP-System)                                          *
* QUSER : Valid user account in target system                 *
* (QAPPL=X1SA)                                               *
* QPASS : User password in target system                      *
* QSTDA : Driver start date                                  *
* QSTTI : Driver start time                                  *
* QSTART : A=Automatic,                                      *
* M=Manual start of driver program                           *
* QANZHL : No. of TS02 transaction entries in the queue       *
***********************************************************************
TABLES    : QPARM.
***********************************************************************
*.............Set queue name..........................................*
  QPARM-QDEST = QDEST.
  QPARM-QAPPL = QAPPL.
  QPARM-QDTYP = 'RODC'.
  QPARM-QMAND = '000'.
*.............Set driver specific parameters..........................
  QPARM-QSTDA = QSTDA.
  QPARM-QSTTI = QSTTI.
  QPARM-QSTRT = QSTART.
  QPARM-QCORR = 'R'.
*.............Set SAP specific data......................*
  QPARM-QDUSR = QUSER.
  QPARM-QDPAS = QPASS.
*.............Open the queue.................................*
OPEN QUEUE QPARM.
*.............Write records in the queue.............*
PERFORM WRITE-QUEUE.
*
Program RSAPPQ10

*.............Close the queue..................*
CLOSE QUEUE QPARM.

*-------------------------------------------------------------*
* Insert records in the queue.                                *
* Sample: Remote ODC data for the transaction TS02.           *
*-------------------------------------------------------------*
FORM WRITE-QUEUE.

DATA:
  TLNR(5)  TYPE C VALUE '00000',
  AREA(42) TYPE C.

FIELD-SYMBOLS: <F>.

DO QANZHL TIMES.
  ADD 1 TO TLNR.

  * Insert first dynpro
  *
  QPARM-QFIRS = 'Y'.
  QPARM-QLAST = 'N'.
  QPARM-QDTRC = 'TS02'.
  QPARM-QDPGM = 'SAPPG02'.
  QPARM-QDDYN = '0041'.
  CLEAR AREA.
  AREA   = '220416 10805160108061601'.
  ASSIGN AREA(38) TO <F>.
  TRANSFER <F> TO QUEUE QPARM.

  * Insert second dynpro
  *
  QPARM-QFIRS = 'N'.
  QPARM-QLAST = 'N'.
  QPARM-QDTRC = 'TS02'.
  QPARM-QDPGM = 'SAPPG02'.
  QPARM-QDDYN = '0060'.
  CLEAR AREA.
  AREA   = '221211 10805160108061601'.
  AREA+17(5) = TLNR.
  ASSIGN AREA(22) TO <F>.
  TRANSFER <F> TO QUEUE QPARM.

  * Insert third dynpro
  QPARM-QFIRS = 'N'.
  QPARM-QLAST = 'Y'.
  QPARM-QDTRC = 'TS02'.
  QPARM-QDPGM = 'SAPPG02'.
  QPARM-QDDYN = '0062'.
  CLEAR AREA.
  AREA   = '180762123456789012'.

  *
ASSIGN AREA(18) TO <F>.
TRANSFER <F> TO QUEUE QPARM.
ENDDO.
*
ENDFORM.
Program RSAPPQ20

**RSAPPQ20**

* Sample program for asynchronous communication via the DOUT file with transfer data taken from the SAP spool file (transfer format SPLD)

**PARAMETERS:**

- **QDEST(8)** TYPE C DEFAULT 'K43 ,'
- **QAPPL(8)** TYPE C DEFAULT 'X1SA ,'
- **QUSER(12)** TYPE C DEFAULT 'RODC-TEST ,'
- **QPASS(8)** TYPE C DEFAULT 'TEST ,'
- **QSTDA** TYPE D DEFAULT SY-DATUM,
- **QSTTI** TYPE T DEFAULT SY-UZEIT,
- **QSTART(1)** DEFAULT 'A ,'
- **QSPLNR(5)** DEFAULT '0'.

**TABLES :**

- **QPARM**.

**DATA :**

- **SPNR** TYPE P,
- **BEGIN OF SPOOL,**
  - **SPMD** LIKE APQD-SPMD VALUE '000',
  - **SPNR** LIKE APQD-SPNR VALUE '00000',
  - **SPPW** LIKE APQD-SPPW VALUE 'PASS',
- **END OF SPOOL**.

**FIELD-SYMBOLS :** <F>.

*....Set queue name....*

- **QPARM-QDEST =** QDEST.
- **QPARM-QAPPL =** QAPPL.
- **QPARM-QDTYP =** 'SPLD'.

April 2001
QPARM-QMAND = '000'.
QPARM-QABAP = 'RSAPPQ40'.
QPARM-QMODU = 'RECEIVE'.
*
*.............Set driver specific parameters...............*
QPARM-QSTDA = QSTDA.
QPARM-QSTTI = QSTTI.
QPARM-QSTRT = QSTART.
QPARM-QCORR = 'R'.
*
*.............Set SAP specific data........................*
QPARM-QDUSR = QUSER.
QPARM-QDPAS = QPASS.
*
*.............Open the queue.........................             *
OPEN QUEUE QPARM.
*
*.............Write record in the queue..............             *
QPARM-QFIRS = 'Y'.
QPARM-QLAST = 'Y'.
SPOOL-SPNR = SPNR = QSPLNR.
ASSIGN SPOOL TO <F>.
TRANSFER <F> TO QUEUE QPARM.
*
*.............Close the queue......................             *
CLOSE QUEUE QPARM.
Program RSAPPQ30

PARAMETERS:

QDEST(8) TYPE C DEFAULT 'K43     ',
QAPPL(8) TYPE C DEFAULT 'X1SA    ',
QUSER(12) TYPE C DEFAULT 'RODC-TEST   ',
QPASS(8) TYPE C DEFAULT 'TEST    ',
QSTDA TYPE D DEFAULT SY-DATUM,
QSTTI TYPE T DEFAULT SY-UZEIT,
QSTART(1) TYPE C DEFAULT 'A'.

PARAMETERS:

QDEST : Connection name of target system
QAPPL : Transaction program name in target system
QUSER : Valid user account in target system
QPASS : User password in target system
QSTDA : Driver start date
QSTTI : Driver start time
QSTART : A=Automatic,
M=Manual start of driver program

TABLES :

QPARM.

QPARM-QDEST = QDEST.
QPARM-QAPPL = QAPPL.
QPARM-QDTPY = 'CPIC'.
QPARM-QMAND = '000'.
QPARM-QABAP = 'RSAPPQ40'.
QPARM-QMODU = 'RECEIVE'.

QPARM-QSTDA = QSTDA.
QPARM-QSTTI = QSTTI.
QPARM-QSTRT = QSTART.
QPARM-QCORR = 'R'.

QPARM-QDUSR = QUSER.
QPARM-QDPAS = QPASS.
* *
* ............Open the queue..........................................*
OPEN QUEUE QPARAM.
* *
* .............Write records in the queue..............................*
PERFORM WRITE-QUEUE.
* *
* .............Close the queue.........................................*
CLOSE QUEUE QPARAM.
* *
* -------------------------------------------------------------*
*       FORM WRITE-QUEUE.                                     *
* -------------------------------------------------------------*
*       Insert records in the queue.                          *
*       Sample : Insert three records of type char.           *
* *-------------------------------------------------------------*

FORM WRITE-QUEUE.
*
DATA:
   AREA(17) TYPE C.
*
FIELD-SYMBOLS: <F>.
*
* Insert first record
*
   QPARAM-QFIRS = 'Y'.
   QPARAM-QLAST = 'N'.
   CLEAR AREA.
   MOVE 'first record' TO AREA.
   ASSIGN AREA(12) TO <F>.
   TRANSFER <F> TO QUEUE QPARAM.
*
* Insert second record
*
   QPARAM-QFIRS = 'N'.
   QPARAM-QLAST = 'N'.
   CLEAR AREA.
   MOVE 'second record' TO AREA.
   ASSIGN AREA(13) TO <F>.
   TRANSFER <F> TO QUEUE QPARAM.
*
* Insert third record
*
   QPARAM-QFIRS = 'N'.
   QPARAM-QLAST = 'Y'.
   CLEAR AREA.
   MOVE 'third record' TO AREA.
   ASSIGN AREA(12) TO <F>.
   TRANSFER <F> TO QUEUE QPARAM.
*
ENDFORM.
APC and APQ Headers DSECT for Assembler Programs

Definition

APC Header as Assembler DSECT

*-------------------------------------------------------------*
*........APC HEADER...........................................*
*-------------------------------------------------------------*
APCDUMMY DSECT
APCREQID DS CL4 REQUEST ID:
*       SAP-CONNECT:           CONN
*       SAP-FREE:              FREE
*       SAP-APPC-COMMUNICATION: APPC
APCTYPE DS CL4 TYPE OF PROCESSING:
*       BATCH INPUT:           BTCI
*       PRESENTATION ON IWS:   DYNP
*       REMOTE DIALOG CALL:    RDIA
*       FREE PROTOCOL:         BLANK
APCMODNR DS CL1 MODE NUMBER
APCSTYPE DS CL1 START NEW TYPE (X)
APCERRCD DS CL1 ERROR-/ABEND-CODE
APCCHSET DS CL1 X'C5' = EBCDIC otherwise ASCII
APCVDATA EQU * PARAMETERS

*........ERROR-MESSAGE (APCERRCD = E).................................*
ORG APCVDATA
APCEMSGN DS CL5 MESSAGE-NUMBER
APCEMSGT DS CL80 MESSAGE-TEXT
APCEMSGL EQU *-APCDUMMY MESSAGE-LENGTH

SPACE 1

*........REQUEST-ID CONN.............................................*
ORG APCVDATA
APCMANDT DS CL3 CLIENT
APCBNAME DS CL12 BATCH-INPUT GROUPNAME (TYPE=BTCI)
*   OR USERNAME
APCPASSW DS CL8 PASSWORD
APCLANGU DS CL1 LANGUAGE
APCKORRV DS CL1 CORRECTION RESPONSIBILITY
(TYPE=BTCI)
*       R = RECEIVER
*       S = SENDER
APCCMSGL EQU *-APCDUMMY MESSAGE-LENGTH

SPACE 1

*........REQUEST-ID CONN.............................................*
*........FREIES PROTOKOLL.............................................*
APCPGMNM DS CL8 ABAP PROGRAM NAME
APCMODNM DS CL30 ABAP MODULE NAME

*........REQUEST-ID APPC.............................................*
*........BATCH INPUT PROCESSING (TYPE = BTCI).........................*
ORG  APCVDATA            HOST INPUT..............................
APCBSYNC DS  CL1                 SYNCPOINT INFORMATION
APCBSYNM EQU  C'S'                  START TRANSACTION
APCBSTAT DS  CL1                 STATE OF PROCESSING
APCBSTAS EQU  C'S'                  SYNCHRONOUS VB
APCBTCOD DS  CL4                 TRANSACTION
APCBFDNM DS  CL8                 PROGRAM NAME
APCBDYNR DS  CL4                 DYNPRO NUMBER
APCBSEFR DS  CL1                 SEPERATOR
APCBCURF DS  CL30                CURSOR-POSITION ON FIELD
APCBMSGL EQU  *-APCDUMMY          BTCI-HEADER LENGTH
DS             CL1
APCBDATA EQU  *                   START OF BDC DATA
SPACE 1
ORG  APCVDATA            RETURN INFORMATION....................
APCSRSTAT DS  CL1                 STATE
APCSRSTAS EQU  C'S'                  START TRANSACTION
APCSRSTAF EQU  C'F'                  TRANSACTION FINISHED
APCSRSTAE EQU  C'E'                  TRANSACTION IN ERROR
APCSRSTAN EQU  C'N'                  GET NEXT MESSAGE
APCSRSTAA EQU  C'A'                  ABEND APC-PROCESSING

APCRCODE DS  CL4                 ABENDCODE
APCRPGMN DS  CL8                 PROGRAM NAME
APCRDYNR DS  CL4                 DYNPRO NUMBER
APCRMNGN DS  CL5                 MESSAGE-NUMBER
APCRMSGT DS  CL80                MESSAGE-TEXT
APCRMMSGL EQU  *-APCDUMMY          MESSAGE-LENGTH
SPACE 1
*........REQUEST-ID APPC..............................................*
*........DIALOG WITH WORKSTATION (TYPE = DYNP)........................*
ORG  APCVDATA
APCDSTAT DS  CL1                 STATE OF PROCESSING
APCDSTAD EQU  C'D'                  SEND DYNPRO (INITIAL OUTPUT)
APCDSTAM EQU  C'M'                  SEND MESSAGE (DIALOGUE)
APCDTCOD DS  CL4                 TRANSAKTION / REPORT RJE03605
APCDDATA EQU  *                   DATA (DYNPRO OR DIALOGUE MESSAGE)
SPACE 1
*........REQUEST-ID APPC..............................................*
*........REMOTE DIALOGCALL (TYPE = RDIA)..............................*
ORG  APCVDATA
APCRPROG DS  CL8                 PROGRAM NAME
APCRDYNP DS  CL4                 DYNPRO NUMBER
APCRDATA EQU  *                   PARAMETER

APQ-Header als Assembler DSECT

*-------------------I-RECORD FOR OPEN GROUP (MAPPE)-------------------*
APC and APQ Headers DSECT for Assembler Programs

*-------------------------------------------------------------*
$SBG3 DS 0C
APQI$BGIN DS 0CL1

***************************************************************
* SEGMENT - HEADER                                         *
***************************************************************
APQILENG DC XL2'176' MAXIMUM SEGMENT LENGTH
APQIRCRES DC XL2'0'
***************************************************************
* KEY                                                       *
***************************************************************
APQIRTYP DC CL1' ' RECORD TYPE ID FOR APPC DATA RECORDS
APQIQNAM DS 0CL(61)
APQIDEST DC CL8' ' SAP TARGET SYSTEM NAME
APQIAAPPL DC CL8' ' APPLIKATIONSNAME
APQIDTYP DC CL4' ' DATA TYPE OF TRANSFERRED DATA UNIT
APQIMAND DC CL3' ' CLIENT IN SAP TARGET SYSTEM
APQIPROG DC CL8' ' SAP PROGRAM IN TARGET SYSTEM
APQIMODP DC CL30' '
APQITRAN DC XL4'0' APPC INTEGER
APQIRECO DC XL4'0' APPC INTEGER
APQIFIXP DS 0CL(44)
APQITREI DC CL8' ' APPC PROGRAM NAME
APQISTRT DC CL1' ' APPC STATUS
APQICORR DC CL1' ' APPC STATUS
APQICRDA DC CL8' '
APQICRTI DC CL6' '
APQIUSER DC CL12' '
APQIPASS DC CL8' ' APPC PASSWORD FOR BTCI DATA TRANSFER
APQIOUAC DC CL1' ' APPC STATUS
APQIOUSY DC CL8' ' APPC PROGRAM NAME
APQIOUAP DC CL8' ' APPC PROGRAM NAME
APQIOUID DC XL4'0' APPC INTEGER
APQIOUTR DC XL4'0' APPC INTEGER
APQIOURE DC XL4'0' APPC INTEGER
APQIOUDA DC CL8' '
APQIOUTI DC CL6' '
APQIOUEN DC CL5' ' APPC MESSAGE NUMBER
APQIOUEM DC CL80' '
APQIINAC DC CL1' ' APPC STATUS
APQIINSY DC CL8' ' APPC PROGRAM NAME
APQIINAP DC CL8' ' APPC PROGRAM NAME
APQIINID DC XL4'0' APPC INTEGER
APQIINRE DC XL4'0' APPC INTEGER
APQIINDA DC CL8' '
APQIINTI DC CL6' '
APQIINEN DC CL5' ' APPC MESSAGE NUMBER
APQIinem DC CL80' '
APQIEND EQU *
Conversion Tables EBCIDC from/to ASCII

The functions CMCNVI and CMCNVO, used to convert between EBCIDC format and ASCII format, work with predefined standard tables. If your application requires other tables, you can create and include your own conversion tables.

Creating Your Own Tables [Seite 220]
Sample File with Conversion Tables [Seite 221]
Creating Your Own Tables

Prerequisites

To do this, you must assign the name of a file, which contains the conversion table, to the shell variable CONVERT before starting the corresponding CPI-C program.

Procedure

When creating a conversion table, you must know the following:

- A comment must begin with the character `*`.
- Blank lines are allowed.
- The ASCII->EBCDIC conversion "a" is defined first, then the EBDIC->ASCII conversion "b".
- The function values are represented without hyphens or blanks as 2-character HEX numbers.
- Only the function values are entered in the file. The arguments are derived from the sequence of function values. The file has the following structure:

```
a(0)a(1)a(2)  a(255)
b(0)b(1)b(2)  b(255)
```

There are no rules for line division.
Sample File with Conversion Tables

* @(#)convert 20.5 SAP 93/04/05
*
* SAP AG Walldorf
* Systems, Applications and Products in Data Processing
*
* (C) Copyright SAP AG 1992
*
********************************
* ASCII -> EBCDIC Conversion   *
********************************
4040404040404040     * 000 - 007
4040404040404040     * 008 - 015
4040404040404040     * 016 - 023
4040404040404040     * 024 - 031
404F7F7B5B6C507D     * 032 - 039
4D5D5C4E6B604B61     * 040 - 047
F0F1F2F3F4F5F6F7     * 048 - 055
F8F97A5E4C7E6E6F     * 056 - 063
B5C1C2C3C4C5C6C7     * 064 - 071
C8C9D1D2D3D4D5D6     * 072 - 079
D7DBD9E2E3E4E5E6     * 080 - 087
E7E8E963ECFC5F6D     * 088 - 095
7981828384858687     * 096 - 103
8889919293949596     * 104 - 111
979899A2A3A4A5A6     * 112 - 119
A7A8A943BBDC5940     * 120 - 127
4040404040404040     * 128 - 135
4040404040404040     * 136 - 143
4040404040404040     * 144 - 151
4040404040404040     * 152 - 159
41AAB0B19FB2CC7C     * 160 - 167
BDB49A8ABA40AFBC     * 168 - 175
908FEAFABEA0B6B3     * 176 - 183
40DA9B8BB7B8B9AB     * 184 - 191
646562664A679E68     * 192 - 199
7471727378757677     * 200 - 207
AC69EDEEEBEFE0BF     * 208 - 215
80DFEFB5AAD8EA1      * 216 - 223
44454246C0479C48     * 224 - 231
5451525358555657     * 232 - 239
8C49CDCEBC6F6AE1     * 240 - 247
70DDDEBD08DAEDF      * 248 - 255
********************************
* EBCDIC -> ASCII Conversion   *
********************************
2020202020202020     * 000 - 007
2020202020202020     * 008 - 015
2020202020202020     * 016 - 023
2020202020202020     * 024 - 031
2020202020202020     * 032 - 039
If you set the shell variable CONVERT to the name `convert` (possibly the complete path if not in the current directory), this file is imported when CMCNVI or CMCNVO are first called and overwrites the standard conversion table.

You can use the SAP-specific functions `SAP_CMLOADCONVTAB` or `SAP_CMMODCONVTAB` to make additional modifications to the conversion table.