Middleware Suite - Application Integration

NCI - Network Computing Interface

SAP R/3 RFC Server Interface
Implementation Guide
Version: 3.1

T - - Systems -

Impressum

Publisher	**T-Systems Enterprise Services GmbH** Computing Services & Solutions (CSS) System Products & Automation (MSY-PA)
Responsible	**T-Systems Enterprise Services GmbH** Computing Services & Solutions (CSS) System Products & Automation (MSY-PA) Fasanenweg 9, 70771 Leinfelden-Echterdingen, Germany cc.middleware@t-systems.com +49 89/1011-4687

Document information

Name of file

NCI SAP R/3 RFC Server Interface

Version / Revision	Date	Revision
This edition relates to NCI version NCI 3.1 • PNCI310/QZ05046 on z/OS • PNCI310/REL1003 on Unix/Windows	25/11/2005 08:49:00	220

List of available NCI documentation:

NCI Application Programming Reference
NCI Installation and Customization for Distributed Systems
NCI Installation and Customization for z/OS and OS/390
NCI MQ File Transfer Utilities
NCI MQ Security Suite
NCI SAP R/3 RFC Server Interface
NCI Additional Features

Additional License Information

Table of Contents

List of Tables

List of Figures

1 Overview

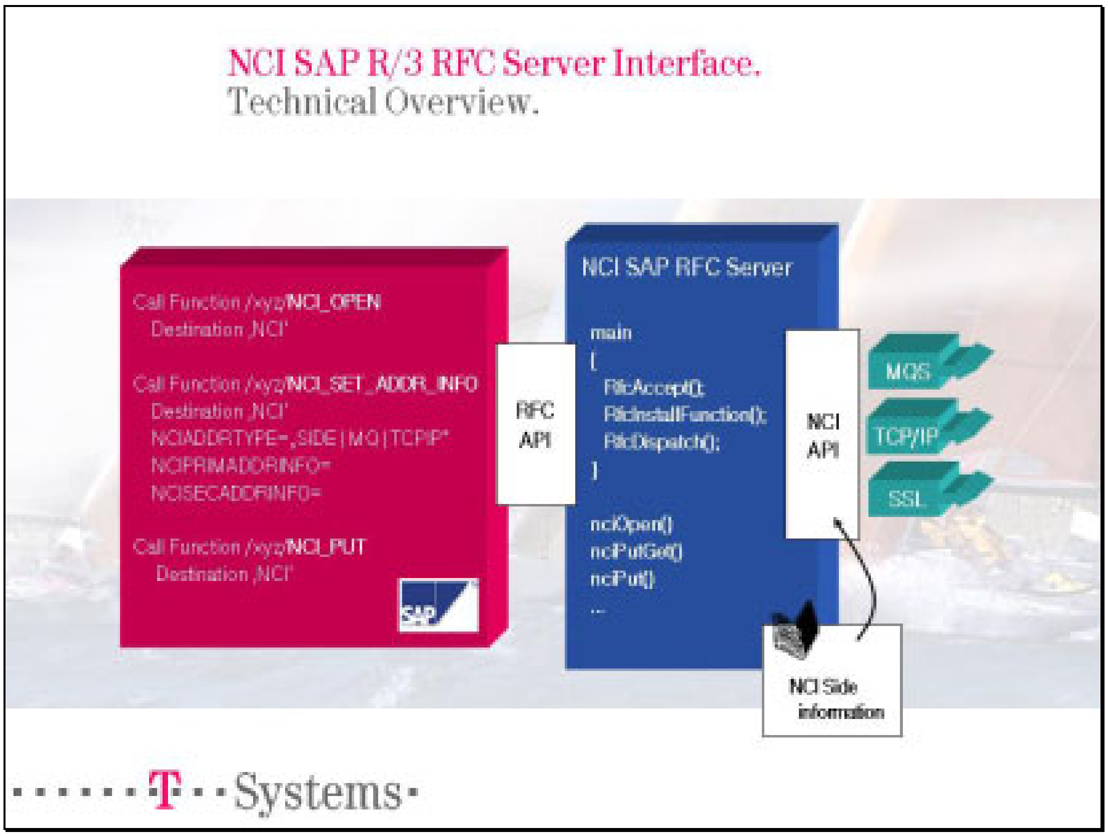
1.1 Generally

The NCI SAP RFC Server Interface provides access to NON-SAP programs from SAP R/3 ABAP programs.
The interface is based on the standard NCI Application Programming Interface and thus almost all features available with NCI can be used. This includes:

- Communication via MQSeries, TCP/IP or SSL.
- Symbolic Addressing via sideinformation
- Connectivity to existing NCI servers

1.2 Functional Overview

The NCI SAP RFC Server Interface uses the SAP RFC API to communicate with the ABAP program. The first call of a NCI function within an ABAP program will start the appropriate NCI RFC server program and the RFC communication will be established. See also Figure 1-1



```
Call Function /xyz/NCI_OPEN
  Destination ,NCI'

Call Function /xyz/NCI_SET_ADDR_INFO
  Destination ,NCI'
  NCIADDRTYPE=,SIDE | MQ | TCPIP"
  NCIPRIMADDRINFO=
  NCISECADDRINFO=

Call Function /xyz/NCI_PUT
  Destination ,NCI'
```

```
main
{
  RfcAccept();
  RfcInstallFunction();
  RfcDispatch();
}

nciOpen()
nciPutGet()
nciPut()
...
```

Figure 1-1: NCI SAP R/3 RFC Server Interface – Technical Overview

2 Installation & Customization

The NCI SAP RFC Server Interface is shipped and installed as a part of the NCI/API package. For more information about installing the NCI/API package refer to the *NCI Application Programming Reference* documentation.

2.1 SAP Customization

Before you can use the NCI SAP RFC Server in an ABAP the following customization steps in the SAP system are required.

1. **Define a RFC destination**
   - o Use SAP transaction SM59
   - o Choose NCI as the name for the RFC Destination.
     This name have to be used in the ABAP programms to address the NCI RFC Server.
   - o
```
o       ...
o       call Function /xxx/NCI_OPEN   destination 'NCI'
o       ...
```
   - o Choose ***:Activation Type Start*** and ***Start on Application server***
   - o The program parameter specifies the NCI SAP RFC Server program **ncisapi0**. The directory depends on where the NCI package was installed.

     Note: Make sure that the NCI RFC Server program **ncisapi0** is able to find the NCI shared libraries at runtime. If the NCI components were not installed in the system libraries the shared library search path must be extended using a platform specific environment variable. For example on SUN Solaris the environment variable LD_LIBRARY_PATH must contain the path where the NCI shared libraries were installed. On other platforms the name of the variable to use may differs.
     The environment variable can be set by updating the startup script of the SAP system or to call the NCI RFC server program **ncisapi0** indirectly via a shell script. See the sample ncisapi0.sh in the NCI samples directory.

2. **Define data elements**
   Use the SAP transaction SE11 to define data elements for the parameters passed between the ABAP and the NCI RFC Server. The following data elements need to be defined:

Z_NCIINT4	Integer length 4
Z_NCICHAR8	Character String length 8
Z_NCICHAR24	Character String length 24
Z_NCICHAR255	Character String length 255
Z_NCICHARxxx	Character String buffer of length xxx. Buffer used to send/receive user data via buffer. See **NCIGet, NCIPut, NCIPutGet**. for more information. The max. length that can be defined is 65.535 Bytes. This length is a restriction of the SAP RFC API.

3. **Define a data structure for NCI**
   Use the SAP transaction SE11 to define a data structure. Choose the name ZNCISRV. Define the following components in this structure.

Component	Component type
HANDLE	Z_NCIINT4
RETURNCODE	Z_NCIINT4
REASONCODE	Z_NCIINT4

Component	Component type
CORRELID	Z_NCICHAR24
MSGID	Z_NCICHAR24
OUTBUFFER	Z_NCICHARxxx
OUTLENGTH	Z_NCIINT4
INBUFFER	Z_NCICHARxxx
INLENGTH	Z_NCIINT4
ADDRTYPE	Z_NCICHAR8
PRIMADDRINFO	Z_NCICHAR255
SECADDRINFO	Z_NCICHAR255
SERVICEID	Z_NCICHAR255
ERRORMSGTEXT	Z_NCICHAR255
APPLID	Z_NCICHAR8
ACCOUNTINFO	Z_NCICHAR255
DATACOMPRESS	Z_NCICHAR8
DATACONV	Z_NCICHAR8
DATAENCRYPT	Z_NCICHAR8
KEEPCONNECTION	Z_NCICHAR8
SYNCPOINTOPT	Z_NCICHAR8
USERID	Z_NCICHAR8
GROUPID	Z_NCICHAR8
PASSWORD	Z_NCICHAR8
NEWPASSWORD	Z_NCICHAR8
RETRYNUMBER	Z_NCIINI4
RETRYTIME	Z_NCIINT4
TIMEOUT	Z_NCIINT4
RFCBUFSIZE	Z_NCIINT4
TABLEROWSIZE	Z_NCIINT4
BUFFERSIZE	Z_NCIINT4

4. Test the functionality using the sample ABAP listed in Appendix A. "ABAP sample" on page 43.

   Note: Before testing do not forget to activate all the definitions.

3 Interface Reference

3.1 Return and Reason Codes

All functions provide a return and a reason code in the export parameters NCIRETURNCODE and NCIREASONCODE. The value of the return code indicates wether the occured error is informational or unrecoverable.
For most errors it is also possible to retrieve a formatted error message using the function call NCIGetErrorMsgText.

Note: To retrieve the formatted error information NCIGetErrorMsgText must be called immediately after the failing function.

3.1.1 Return Codes

Return Code	Description
0	NCI_RC_OK Successful completion
4	NCI_RC_INFO Information, partial completion. Check reason code.
8	NCI_RC_WARNING Warning (currently not used !)
12	NCI_RC_HANDLE Error (NCI handle is missing or is invalid)
16	NCI_RC_ERROR Error (unrecoverable error)

Table 3-1: Return Codes

3.1.2 Reason Codes

Reason codes are always equal to the 4-digit message number of the correspondung formatted message text. Applications should always check the reason code if the NCI function returns a return code of NCI_RC_INFO. For all other errors the application should use the function NCIGetErrorMsgText to retrieve the formatted error message text and save the message to a log file for later error analysis.
The reason codes a function returns in case of NCI_RC_INFO are described in detail, in the appropriate function description chapter.

3.1.3 Message Format

Error Message-Format: **NCIxxxxy ModuleName - ApplicatonID: Message Text**

	Description
xxxx	4-digit Message-Number (equal to the error Reason Code).
y	Error-valence:

- I Information
- W Warning
- E Error

ModuleName	Name of the NCI routine that detected the error.
ApplicationId	Application Identifier set by the application via function *SetApplicationId*.
Message Text	Information about the Error Condition

3.2 Function Reference

The NCI SAP RFC Server Interface provides almost all of the functionality available with the NCI native API. This includes addressing via sideinformation file or environment variables.
To improve performance some functions of the NCI native API were put together into a single call. Currently all available functions are contained in the NCI RFC server program **ncisapi0**. To use the functions you have to define a destination definition in your SAP system for ncisapi0. See "SAP Customization" on page 9 for more information about defining a destination.

By default all the functions are available in the namespace /DBN/. However you can change the used namespace by invoking the NCI RFC server program **ncisapi0** with the *NAMESPACE=* parameter. Please refer to "ncisapi0 - Setting Line Command Parameters" on page 11 for more information about specifying line command parameters for ncisapi0.

3.2.1 NCIBack - Backout Changes

3.2.1.1 Functionname

/DBN/NCI_BACK

3.2.1.2 Description

Backout all changes since the last syncpoint.

Note: Syncpoint processing is only supported for protocol MQSeries.

<u>Behaviour for protocol MQSeries:</u>
The **NCIBack** call indicates the queue manager that all of the message gets and puts occurred since the last syncpoint are to be backed out. Messages put as part of a unit of work are deleted; messages retrieved as part of a unit of work are reinstated on the queue.

For additional information refer to "NCISetOptions - Set additional processing Options" on page 35 parameter *NCISYNCPOINTOPT*.

3.2.1.3 Parameters

<u>Importing</u>

Parameter	Format	Typ
NCIHANDLE	RFCTYPE_INT	Required

<u>Exporting</u>

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
Handle provided by function **NCIOpen**

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

3.2.1.4 Return Codes

Code	Description
0	Successful completion
12	Error (NCI handle is missing or invalid)
16	Error (unrecoverable error)

3.2.2 NCIClose- Cleanup NCI Environment

3.2.2.1 Functionname

/DBN/NCI_CLOSE

3.2.2.2 Description

Closes all open connections for this handle and releases resources obtained for this handle.
NCIClose has to be called for each handle obtained using **NCIOpen**. After closing a handle, the handle is no longer useable!

3.2.2.3 Parameters

Importing

Parameter	Format	Typ
NCIHANDLE	RFCTYPE_INT	Required

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
Handle provided by function **NCIOpen**

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

3.2.2.4 Return Codes

Code	Description
0	Successful completion
12	Error (NCI handle is missing or invalid)
16	Error (unrecoverable error)

3.2.3 NCICmit - Commit Changes

3.2.3.1 Functionname

/DBN/NCI_CMIT

3.2.3.2 Description

Commit all changes since the last syncpoint.

Note: Syncpoint processing is only supported for protocol MQSeries.

Behaviour for protocol MQSeries:
The **NCICmit** call indicates to the queue manager that the application has reached a syncpoint, and that all of the message gets and puts that have occurred since the last syncpoint are to be made permanent. Messages put as part of a unit of work are made available to other applications; messages retrieved as part of a unit of work are deleted.

For additional information refer to "NCISetOptions - Set additional processing Options" on page 35 parameter *NCISYNCPOINTOPT*.

3.2.3.3 Parameters

Importing

Parameter	Format	Typ
NCIHANDLE	RFCTYPE_INT	Required

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
Handle provided by function **NCIOpen**

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

3.2.3.4 Return Codes

Code	Description
0	Successful completion
12	Error (NCI handle is missing or invalid)

16	Error (unrecoverable error)

3.2.4 NCIFreeConnection - Free Connection

3.2.4.1 Functionname

/DBN/NCI_FREE_CONNECTION

3.2.4.2 Description

The effect of NCIFreeConnection depends on the underlying protocol.

Behaviour for protocol TCP/IP:
Closes the connection to the server (if one exists).

Behaviour for protocol MQSeries:
Not supported.

For additional information refer to "NCISetOptions - Set additional processing Options" on page 35 parameter *NCIKEEPCONNECTION*.

3.2.4.3 Parameters

Importing

Parameter	Format	Typ
NCIHANDLE	RFCTYPE_INT	Required

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
Handle provided by function NCIOpen

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

3.2.4.4 Return Codes

Code	Description
0	Successful completion
12	Error (NCI handle is missing or invalid)
16	Error (unrecoverable error)

3.2.5 NCIGet - Get a Message

3.2.5.1 Functionname

/DBN/NCI_GET

3.2.5.2 Description

Note: NCIGet can be only used with protocol MQSeries.

NCIGet receives a message. After **NCIGet** has completed, the application has to examine the NCI Return and Reason Code to check wether the sender of the message waits for a reply.
Depending on the message length, messages can be either received using the buffer specified with parameter *NCIINBUFFER* or the table *NCIINTAB*. Receiving of messages using *NCIINBUFFER* is limited to a maximum message size of 65.535 Bytes. To receive longer messages you have to use the table *NCIINTAB*. In this case the received message is split into several table rows.

3.2.5.3 Parameters

Importing

Parameter	Format (length)	Typ
NCIHANDLE	RFCTYPE_INT	Required
NCIINLENGTH	RFCTYPE_INT	Optional
NCIMSGID	RFCTYPE_BYTE (24)	Optional
NCICORRELID	RFCTYPE_BYTE (24)	Optional

Exporting

Parameter	Format (length)
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT
NCIINBUFFER	RFCTYPE_BYTE (max. 65.535)
NCIINLENGTH	RFCTYPE_INT
NCIMSGID	RFCTYPE_BYTE (24)
NCICORRELID	RFCTYPE_BYTE (24)

Tables

Parameter	
NCIINTAB	Optional

NCIHANDLE
Handle provided by **NCIOpen**.

NCIINLENGTH
This parameter is used for import and export. On import it defines the max. accepted length (in bytes) of a message. On export the real length of the received message is returned. If the parameter is omitted the length of the internally used receive buffer will be used. This length is, by default, 100K Bytes. See "NCISetAttributes - Set Attributes used by NCI RFC server" on page 33 how the default length can be changed.

If *NCIINBUFFER* is used to receive the message *NCIINLENGTH* must not exceed the length of the provided *NCIINBUFFER*

NCIINBUFFER
Buffer to receive the message. The size of a message that can be received using the *NCIINBUFFER* is limited to 65.535 Bytes.
To receive longer messages omit this parameter and use the table *NCIINTAB*.

NCIMSGID
Parameter can be used for import and export. Use this parameter as import parameter to receive a message with the corresponding MQSeries message id.
On export it contains the MQSeries message id of the received message.

NCICORRELID
Parameter can be used for import and export. Use this parameter as import parameter to receive a message with the corresponding MQSeries correlation id.
On export it contains the MQSeries correlation id of the received message.

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

NCIINTAB
To receive a message into a table, *NCIINTAB* has to be specified. The received message is split into multiple table rows that are added to the table. **NCIGet** uses a default rowlength of 255 Bytes to split the message. The default can be changed using the function **NCISetAttributes** (see "NCISetAttributes - Set Attributes used by NCI RFC server" on page 33).
NCIINTAB must be omitted if a message should be received in *NCIINBUFFER*.

- **Return Codes**

Code	Description
0	Successful completion
4	Information (partial completion)
12	Error (NCI handle is missing or is invalid)
16	Error (unrecoverable)

3.2.5.4 Reason Codes

Return Code = 4 (NCI_RC_INFO)

Reason	Description
28	NCI_RCC_TRUNCATED_MSG the message buffer provided by the application was to small. The received data has been truncated. The entire length of the message is returned in the NCIINLENGTH export parameter.
32	NCI_RCC_GET_INHIBITED get requests are temporary inhibited (protocol MQSeries only).
36	NCI_RCC_REPLY_REQUIRED Received message requires a reply message.
112	NCI_RCC_SEC_USERID_REQUIRED

	Userid is required to allow access.
116	NCI_RCC_SEC_PWD_REQUIRED
	Userid and password are required to allow access.
120	NCI_RCC_SEC_USERID_NOT_DEFINED
	The specified userid is not defined on server side.
128	NCI_RCC_SEC_USERID_REVOKED
	The specified userid has been revoked on server side.
132	NCI_RCC_SEC_PWD_EXPIRED
	The specified password has expired.
136	NCI_RCC_SEC_PWD_INVALID
	The specified password is invalid.
144	NCI_RCC_SEC_UNKNOWN_DENY
	An unexpected error occured on server side. Request will be denied.
160	NCI_RCC_TIMEOUT
	Timeout occured, no more suitable messages are available.

The reason codes related to security can only be obtained in the case of an MQSeries client in conjunction with the NCI security exits for MQSeries.

3.2.6 NCIGetErrorMsgText - Get Error Message Text

3.2.6.1 Functionname

/DBN/NCI_GET_ERROR_MSG_TEXT

3.2.6.2 Description

If a NCI function fails (*NCIRETURNCODE* not zero) detailed error information can be retrieved using **NCIGetErrorMsgText.** NCIGetErrorMsgText returns a formatted error message text that describes the error.

3.2.6.3 Parameters

Importing

Parameter	Format	Typ
NCIHANDLE	RFCTYPE_INT	Required

Exporting

Parameter	Format (length)
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT
NCIERRORMSGTEXT	RFCTYPE_CHAR (255)

NCIHANDLE
Handle provided by function **NCIOpen**

NCIRETURNCODE
Return Code

NCIREASONCODE
Reason Code

NCIERRORMSGTEXT
Formatted error message text.

3.2.6.4 Return Codes

Code	Description
0	Successful completion
12	Error (NCI handle is missing or invalid)
16	Error (unrecoverable error)

3.2.7 NCIOpen - Initialize NCI Environment

3.2.7.1 Functionname

/DBN/NCI_OPEN

3.2.7.2 Description

Initializes the NCI environment and returns a handle that has to be passed to all subsequent function calls. The open call must be the first call in your application. All the other function calls rely on the successfull open of a NCI handle.

3.2.7.3 Parameters

Exporting

Parameter	Format
NCIHANDLE	RFCTYPE_INT
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
If **NCIOpen** is sucessfull a handle is returned that must be passed as input to all subsequent function calls.

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

3.2.7.4 Return Codes

Code	Description
0	Successful completion
16	Error (unrecoverable)

3.2.8 NCIPut - Put a Message

3.2.8.1 Functionname

/DBN/NCI_PUT

3.2.8.2 Description

Note: NCIPut is only supported with protocol MQSeries.

NCIPut sends a datagram message or a reply message depending on the previous data flow. If the previously received message requested a reply this message is treated as the reply message. In all other cases a datagram message is created.
Messages can be either passed using the buffer *NCIOUTBUFFER* or the table *NCIOUTTAB*. To send a message via buffer, *NCIOUTBUFFER* must contains the message data and *NCIOUTLENGTH* must contain the the length of the message. However sending data using a buffer is limited to 65.535 Bytes. For longer messages the table *NCIOUTTAB* has to be used.
To pass a message via table, NCIOUTTAB must be specified. When using a table **NCIPut** accumulates all rows contained in the table into one single message. The overall length is only limited to the maximum MQSeries message length.

3.2.8.3 Parameters

Importing

Parameter	Format (length)	Typ
NCIHANDLE	RFCTYPE_INT	Required
NCIOUTBUFFER	RFCTYPE_BYTE (max. 65.535)	Optional
NCIOUTLENGTH	RFCTYPE_INT	Optional
NCIMSGID	RFCTYPE_BYTE (24)	Optional
NCICORRELID	RFCTYPE_BYTE (24)	Optional

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

Tables

Table Name	Typ
NCIOUTTAB	Optional

NCIHANDLE
Handle provided by **NCIOpen**.

NCIOUTBUFFER
Buffer containing the data to be sent. The length of the buffer is limited to 65.535 Bytes. For longer messages omit this parameter and use the table *NCIOUTTAB*.

NCIOUTLENGTH
Length of the data contained in the buffer *NCIOUTBUFFER*. The length can be smaller than the total length of *NCIOUTBUFFER* if you only want to send a part of the buffer.
Parameter can be omitted if a table is used to pass the data.

NCIMSGID
Use this parameter to assign a message id to the MQSeries message. If not used, MQSeries automatically creates a message id for every message.

NCICORRELID
Use this parameter to assign a correlation id to the MQSeries message.

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

NCIOUTTAB
Table containing the rows to be sent.
All table rows, by default, must have a length of 255 Bytes The default row length can be changed using the function **NCISetAttributes** (see "NCISetAttributes - Set Attributes used by NCI RFC server" on page 33).
NCIOUTTAB must be omitted if a message is passed via *NCIOUTBUFFER*.

3.2.8.4 Return Codes

Code	Description
0	Successful completion
4	Information (partial completion)
12	Error (NCI handle is missing or is invalid)
16	Error (unrecoverable)

3.2.8.5 Reason Codes

If the *NCIPut* call completes with a return code of 4 (NCI_RC_INFO), the reason code, that can be obtained via function call *NCIGetErrorReasonCode*, provides further information as described below.

Reason	Description
112	NCI_RCC_SEC_USERID_REQUIRED Userid is required to allow access.
116	NCI_RCC_SEC_PWD_REQUIRED Userid and password are required to allow access.
120	NCI_RCC_SEC_USERID_NOT_DEFINED The specified userid is not defined on server side.
128	NCI_RCC_SEC_USERID_REVOKED The specified userid has been revoked on server side.
132	NCI_RCC_SEC_PWD_EXPIRED The specified password has expired.
136	NCI_RCC_SEC_PWD_INVALID The specified password is invalid.
144	NCI_RCC_SEC_UNKNOWN_DENY An unexpected error occured on server side. Request will be denied.

176	NCI_RCC_PRIORITY_EXCEEDS_MAX The priority assigned to a message is greater than the max priority supported by the queue manager. The message is accepted by the queue manager, but placed on the queue at the queue manager's maximum priority.

The reason codes related to security can only be obtained in the case of an MQSeries client in conjunction with the NCI security exits for MQSeries.

3.2.9 NCIPutGet - Put a Message and wait for Reply

3.2.9.1 Functionname

/DBN/NCI_PUT_GET

3.2.9.2 Description

NCIPutGet combines the NCIPut and NCIGet function calls. **NCIPutGet** sends a message and waits for a reply message. The function doesn't return before the reply message has been received or a specified timeout elapsed.
For a description of how to use the parameters to pass and receive messages please refer to "NCIPut - Put a Message" on page 23 and "NCIGet - Get a Message" on page 18.

NCIPutGet supports the protocols TCP/IP and MQSeries.

Special behaviour for protocol MQSeries:
The **NCIPutGet** function call puts a message on a local or remote queue and expects the reply message on either a permanent or dynamic local queue. The processing is as follows:

1. If a local queue exists with the same name as the original queue appended by ".R", this queue will be used as reply queue.
2. If a model queue exists with the same name as the original queue appended by ".R", a dynamic queue based on this model queue will be created and used as reply queue.
3. Otherwise a dynamic queue based on SYSTEM.DEFAULT.MODEL.QUEUE will be created and used as reply queue.

Special behaviour for protocol TCP/IP:
When used with TCP/IP you can communicate with a NCI Server application. For this case it is required to set the parameter *NCISERVICEID* with the **NCISetAddrInfo** function call.

3.2.9.3 Parameters

Importing

Parameter	Format (length)	Typ
NCIHANDLE	RFCTYPE_INT	Required
NCIINLENGTH	RFCTYPE_INT	Optional
NCIOUTBUFFER	RFCTYPE_BYTE (max. 65.535)	Optional
NCIOUTLENGTH	RFCTYPE_INT	Optional

Exporting

Parameter	Format (length)
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT
NCIINBUFFER	RFCTYPE_BYTE (max. 65.535)
NCIINLENGTH	RFCTYPE_INT

Tables

Parameter	
NCIOUTTAB	Optional
NCIINTAB	Optional

NCIHANDLE
Handle provided by **NCIOpen**.

NCIINLENGTH
This parameter is used for import and export. On import it defines the max. accepted length (in bytes) of a message. On export the real length of the received message is returned. If the parameter is omitted the length of the internally used receive buffer will be used. This length is, by default, 100K Bytes. See "NCISetAttributes - Set Attributes used by NCI RFC server" on page 33 how the default length can be changed.
If *NCIINBUFFER* is used to receive the message *NCIINLENGTH* must not exceed the length of the provided *NCIINBUFFER*

NCIINBUFFER
Buffer to receive the message. The size of a message that can be received using the *NCIINBUFFER* is limited to 65.535 Bytes.
To receive longer messages omit this parameter and use the table *NCIINTAB*.

NCIOUTBUFFER
Buffer containing the data to be sent. The length of the buffer is limited to 65.535 Bytes For longer messages omit this parameter and use the table *NCIOUTTAB*.

NCIOUTLENGTH
Length of the data contained in the buffer *NCIOUTBUFFER*. The length can be smaller than the total length of *NCIOUTBUFFER* if you only want to send a part of the buffer.
Parameter can be omitted if a table is used to pass the data.

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

NCIOUTTAB
Table containing the rows to be sent.
All table rows, by default, must have a length of 255 Bytes The default row length can be changed using the function **NCISetAttributes** (see "NCISetAttributes - Set Attributes used by NCI RFC server" on page 33).
NCIOUTTAB must be omitted if a message is passed via *NCIOUTBUFFER*.

NCIINTAB
To receive a message into a table, *NCIINTAB* has to be specified. The received message is split into multiple table rows that are added to the table. **NCIGet** uses a default rowlength of 255 Bytes to split the message. The default can be changed using the function **NCISetAttributes** (see).
NCIINTAB must be omitted if a message should be received in *NCIINBUFFER*.

3.2.9.4 Return Codes

Code Description

0	Successful completion
4	Information (partial completion)
12	Error (NCI handle is missing or is invalid)
16	Error (unrecoverable)

3.2.9.5 Reason Codes

If the *NCIPutGet* call completes with a return code of 4 (NCI_RC_INFO), the reason code, that can be obtained via function call *NCIGetErrorReasonCode*, provides further information as described below.

Reason	Description
28	NCI_RCC_TRUNCATED_MSG The input buffer provided by the application (InBuffer) was to small. The received data has been truncated.
32	NCI_RCC_GET_INHIBITED Get requests are temporary inhibited (protocol MQSeries only).
104	NCI_RCC_TP_NOT_DEFINED The requested TP-Name requested in SERVICEID is not defined on server side.
108	NCI_RCC_SERVER_IS_TERMINATING The requested server has order to stop and is currently in termination phase.
112	NCI_RCC_SEC_USERID_REQUIRED To allow access a userid is required.
116	NCI_RCC_SEC_PWD_REQUIRED To allow access userid and password are required.
120	NCI_RCC_SEC_USERID_NOT_DEFINED The specified userid is not defined on server side.
124	NCI_RCC_SEC_GRPID_NOT_DEFINED The specified groupid is not defined on server side.
128	NCI_RCC_SEC_USERID_REVOKED The specified userid has been revoked on server side.
132	NCI_RCC_SEC_PWD_EXPIRED The specified password has expired. Password and new password are required.
136	NCI_RCC_SEC_PWD_INVALID The specified password is invalid.
140	NCI_RCC_SEC_NEWPWD_INVALID The new specified new password doesn't conform with the password rules defined on server side.
144	NCI_RCC_SEC_UNKNOWN_DENY An unexpected error occured on server side. Request will be denied.
148	NCI_RCC_TP_ALLINUSE The requested TP is not available because all TP instances are in use. This is an unexpected error and should never occur.
160	NCI_RCC_TIMEOUT Timeout occured, no more suitable messages are available.
164	NCI_RCC_ROOT_ACCESS_DENIED A client tries to connect to a server with a userid that has root-authority on the server platform. However the requested server does not allow root access from the client host.
168	NCI_RCC_SESS_FEATURES_RECV_NOTSUP The creator of the message used features (like encryption or conversion) that are not supported from receiver.
176	NCI_RCC_PRIORITY_EXCEEDS_MAX The priority assigned to a message is greater than the max priority supported by the queue manager. The message is accepted by the queue manager, but placed on the queue at the queue manager's maximum priority.

180	NCI_RCC_SEC_NOT_AUTHORIZATION_TO_ACCESS_TP The client user is not authorized to access the requested TP.

<u>Behaviour for protocol MQSeries:</u>
The reason codes related to security can only be obtained in the case of an MQSeries client in conjunction with the NCI security exits for MQSeries.

3.2.10 NCISetAddrInfo - Set Addressing Information

3.2.10.1 Functionname

/DBN/NCI_SET_ADDR_INFO

3.2.10.2 Description

Setup addressing information needed to contact the remote application. The type of information that must be provided depends on the used addressing type (see parameter *NCIADDRTYPE*).

3.2.10.3 Parameters

Importing

Parameter	Format (length)	Typ
NCIHANDLE	RFCTYPE_INT	Required
NCIADDRTYPE	RFCTYPE_CHAR (8)	Required
NCIPRIMADDRINFO	RFCTYPE_CHAR (255)	Required
NCISECADDRINFO	RFCTYPE_CHAR (255)	Required
NCISERVICEID	RFCTYPE_CHAR (255)	Optional

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
Handle provided by **NCIOpen**.

NCIADDRTYPE
The following addressing types are supported:

Type	Description
SIDE	Symbolic addressing via sideinformation. The parameters *NCIPRIMADDRINFO* and *NCISECADDRINFO* will be used to perform symbolic address translation. *NCIPRIMADDRINFO* must provide the name of the sideinformation file *NCISECADDRINFO* must provide the SymbolicName that must have been defined within the sideinformation file. The SymbolicName. will be used • as a logical name to address a physical target system/application (e.g. hostname, portnumber, LU-name, queue manager, queue name). • to provide applicationspecific control options (e.g. Timeout values / Trace options / ...) outside the application.
ENV	Symbolic addressing via environment variables. The parameters *NCIPRIMADDRINFO* and *NCISECADDRINFO* will be used to perform symbolic address translation. *NCIPRIMADDRINFO* must provide the name of the environment variable *NCISECADDRINFO* must provide the SymbolicName that must have been defined within the environment variable. The SymbolicName will be used

- as a logical name to address a physical target system/application (e.g. hostname, portnumber, LU-name, queue manager, queue name).
- to provide applicationspecific control options (e.g. Timeout values / Trace options / ...) outside the application.

TCPIP	TCP/IP socket interface will be used as transport protocol. The parameters *NCIPRIMADDRINFO* and *NCISECADDRINFO* will be used to address the TCP/IP target system and application. *NCIPRIMADDRINFO* must provide the TCP/IP hostname or the IP-address in dotted format (e.g. 53.113.127.10). *NCISECADDRINFO* must provide the TCP/IP servicename (defined in /etc/services) or portnumber (e.g.3450).
MQ	MQSeries will be used as transport protocol. The parameters *NCIPRIMADDRINFO* and *NCISECADDRINFO* will be used to address the MQSeries queue manager and queue name. *NCIPRIMADDRINFO* must provide the name of the MQSeries queue manager. *NCISECADDRINFO* must provide the MQSeries queue name.

NCIPRIMADDRINFO
The meaning of the parameter *NCIPRIMADDRINFO* depends on the setting of parameter *NCIADDRTPYE*

NCIADDRTYPE	Meaning of NCIPRIMADDRINFO
SIDE	*NCIPRIMADDRINFO* must provide the name of the NCI sideinformation file.
ENV	*NCIPRIMADDRINFO* must provide the name of the environment variable.
TCPIP	*NCIPRIMADDRINFO* must provide the TCP/IP hostname or the IP-address in dotted format (e.g. 53.113.127.10).
MQ	*NCIPRIMADDRINFO* must provide the name of the MQSeries queue manager, otherwise the default queue manager will be used.

NCISECADDRINFO
The meaning of the parameter *NCISECADDRINFO* depends on the setting of parameter *NCIADDRTPYE*

NCIADDRTYPE	Meaning of NCISECADDRINFO
SIDE	SymbolicName defined within the sideinformation file.
ENV	SymbolicName that must have been defined within the environment variable.
TCPIP	TCP/IP servicename (defined in /etc/services) or portnumber (e.g.3450).
MQ	name of MQSeries queue.

NCISERVICEID
This parameter is optional and depends on the used protocol TCP/IP or MQ.

Behaviour for protocol TCP/IP:
The *NCISERVICEID* is only used by function **NCIPutGet**, to identify the transaction program (TP-Name) that should be started by the NCI communication manager to handle this request.
Note: Only the first 60 bytes will be used as TP-Name.

Behaviour for protocol MQSeries:
On function **NCIPut:** and **NCIPutGet** the ServiceId can be used to assign a message identifier to the message being sent.
On function **NCIGet:** the ServiceId can be used as a search argument. Only messages with the requested message identifier (ServiceId) will be retrieved.
Note: Only the first 28 bytes will be used as message identifier.

NCIRETURNCODE
Return code

NCIREASONCODE
Reason code

3.2.10.4 Return Codes

Code	Description
0	Successful completion
12	Error (NCI handle is missing or invalid)
16	Error (unrecoverable error)

3.2.11 NCISetAttributes - Set Attributes used by NCI RFC server

3.2.11.1 Functionname

/DBN/NCI_SET_ATTRIBUTES

3.2.11.2 Description

The NCI RFC server program uses some default values concerning the functions **NCIPut, NCIGet** and **NCIPutGet**. **NCISetAttributes** can be used to change these defaults.

Currently the following defaults are used.

- *NCIRFCBUFSIZE* (default: 65.535 Bytes)
  Length in bytes of the buffer used to pass data between ABAP and NCI RFC server program. It is no longer recommended to change this value, because there is no need to change it. By default a value of 65.535 Bytes is assumed. This default represents the max. size of data that can be echanged between an ABAP and a RFC server using a simple data type. To send or receive longer messages you have to work with SAP tables.
- *NCITABLEROWSIZE* (default: 255 Bytes)
  Length of SAP table row.
- *NCIBUFFERSIZE* (default: 100K Bytes)
  Length in bytes of an internally used buffer to receive and send messages.

Changing one of the default values has a global scope. These values are not bound to a specific handle. All other subsequent NCI function calls are affected by the change.

3.2.11.3 Parameters

Importing

Parameter	Format	Typ
NCIRFCBUFSIZE	RFCTYPE_INT	Optional
NCITABLEROWSIZE	RFCTYPE_INT	Optional
NCIBUFFERSIZE	RFCTYPE_INT	Optional

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIRFCBUFSIZE
Buffer length used for NCIINBUFFER and NCIOUTBUFFER.

NCITABLEROWSIZE
Change this value to send or receive tables that have another table row length as the default of 255 Bytes.

NCIBUFFERSIZE
Length in bytes of an internally used buffer to send or receive messages. This parameter is usually not of interest, because if this buffer is too small to satisfy a request (e.g. NCIGet specifies NCIINLENGTH > 100K Bytes) a new buffer is allocated and freed when the function returns.
However, in some situations it makes sense to change the buffer size. For example if you intend to send or receive a series of messages exceeding the default buffer length of 100K Bytes you should increase this value to omit allocating and freeing the buffer for every request.

3.2.11.4 Return Codes

Code	Description
0	Successful completion
16	Error (unrecoverable)

3.2.12 NCISetOptions - Set additional processing Options

3.2.12.1 Functionname

/DBN/NCI_SET_OPTIONS

3.2.12.2 Description

Use this function to set additional processing options.
Almost all SET functions of the NCI native interface were put together into this single function call to improve performance.
All parameters are optional. If a parameter is not used its value will be unchanged.

Note: The function **NCISetAddrInfo** resets all processing options to default values. To work correctly **NCISetOptions** should be called after **NCISetAddrInfo**.

3.2.12.3 Parameters

Importing

Parameter	Format	Typ
NCIHANDLE	RFCTYPE_INT	optional
NCIAPPLID	RFCTYPE_CHAR(8)	optional
NCIACCOUNTINFO	RFCTYPE_CHAR(255)	optional
NCIDATACOMPRESS	RFCTYPE_CHAR(8)	optional
NCIDATACONV	RFCTYPE_CHAR(8)	optional
NCIDATAENCRYPT	RFCTYPE_CHAR(8)	optional
NCIKEEPCONNECTION	RFCTYPE_CHAR(8)	optional
NCISYNCPOINTOPT	RFCTYPE_CHAR(8)	optional
NCIUSERID	RFCTYPE_CHAR(8)	optional
NCIGROUPID	RFCTYPE_CHAR(8)	optional
NCIPASSWORD	RFCTYPE_CHAR(8)	optional
NCINEWPASSWORD	RFCTYPE_CHAR(8)	optional
NCIRETRYNUMBER	RFCTYPE_INT	optional
NCIRETRYTIME	RFCTYPE_INT	optional
NCITIMEOUT	RFCTYPE_INT	optional

Exporting

Parameter	Format
NCIRETURNCODE	RFCTYPE_INT
NCIREASONCODE	RFCTYPE_INT

NCIHANDLE
Handle provided by function **NCIOpen**

NCIAPPLID
The application identifier is used to identify the originator of an NCI error message. NCI error messages

always include the Application Identifier.
For a description of NCI error message format refer to chapter: "Return and Reason Codes" on page 11.

NCIACCOUNTINFO
Note This feature is only supported with protocol: MQSeries.

Associates accounting information with the message(s) being sent. The receiver of a message can access the account information by function call: *NCIGetAccountInfo*.

NCIDATACOMPRESS

Controls if message data should be compressed before it is transmitted.

Valid options are:

State	Description
YES	Compress userdata.
<u>NO</u>	No data compression will be done.

NCIDATACONV
Controls if message data should be converted from ASCII to EBCDIC and vice versa. Data conversion is only possible for valid character strings (message data must be present in character format).

A client application may request message data conversion by setting *NCIDATACONV* to YES.
Example: If a client application (running under Unix or Windows) sends message data in ASCII and option *NCIDATACONV(YES)* is in use, the data will be converted to EBCDIC, before the server application (running under OS/390) gets control (after function nciGet).
If the client wants to receive the return data in ASCII the server application must set the data conversion option to SAME or YES.

<u>Behaviour for protocol TCP/IP:</u>
NCI makes internal use of XDR (eXternal Data Representation), which is the standard TCP/IP feature to convert data from host to network format and backward.

<u>Behaviour for protocol MQSeries:</u>
Data conversion will be done by MQSeries conversion routines.

Valid options are:

State	Description
YES	Convert userdata (ASCII/EBCDIC).
<u>NO</u>	No data conversion will be done.
SAME	Conversion of userdata depends on the client request.

NCIDATAENCRYPT
Note: This feature is only supported with protocol TCP/IP.

Controls if message data should be encrypted before transmission. NCI uses an internal DES algorithm for message data encryption (message data must be present in character format). Valid options are:

State Description

YES	Encrypt userdata (via DES algorithm).
NO	No data encryption will be done.

NCIKEEPCONNECTION
Default value for protocol TCP/IP is **NO**.
Default value for protocol MQSeries is **YES**.

The effect of the parameter *NCIKEEPCONNECTION* depends on the used protocol.

Behaviour for protocol TCP/IP:
Controls whether a connection to the NCI server should be kept open or reed immediately after the function call **NCIPutGet** has executed. Keeping a connection open is useful to perform a multi-staging conversation to the same instance of a server program or to reduce processing overhead (no connect/disconnect for every individual request). The connection will be automatically freed by the next function call **NCIPutGet** to a different partner, by the function call **NCIFreeConnection** or by the function call **NCIClose.**.

Behaviour for protocol MQSeries:
Controls whether queues are left open after executing the function calls **NCIPut**, **NCIGet** and **NCIPutGet** to reduce processing overhead. These function calls implicitly try to open the respective queue, if it is not already open, when processing a message. Open Queues are implicitly closed when **NCIClose.** is called.

Valid options are:

Value	**Description**
YES	Connections will be kept.
NO	Connections will be freed after each request.

NCISYNCPOINTOPT
Note: Syncpoint processing is only supported for protocol MQSeries.

Requests can be made with or without syncpoint control. A unit of work can be committed with the function *nciCmit* or backed out with the function *nciBack*.

Valid syncpoint options are:

Option **Description**
YES *Protocol MQSeries*

- NCIPut
  Put message with syncpoint control. The request is to operate within the normal unit-of-work protocols. The message is not visible outside the unit of work until the unit of work is committed. If the unit of work is backed out, the message is discarded.
- NCIGet
  Get message with syncpoint control. The request is to operate within the normal unit-of-work protocols. The message is marked as being unavailable to other applications, but it is deleted from the queue only when the unit of work is committed. The message is made available again if the unit of work is backed out.

NO *Protocol MQSeries*

- NCIPut
  Put message without syncpoint control. The request is to operate outside the normal unit-of-work protocols. The message is available immediately, and it cannot be deleted by backing out a unit of work.
- NCIGet
  Get message without syncpoint control. The request is to operate outside the normal unit-of-work protocols. The message is deleted from the queue immediately (unless this is a browse request). The message cannot be made available again by backing out a unit of work.

Note: By default, in order to make applications portable, syncpoint processing is turned off on all platforms. This is different than for native MQSeries programs on z/OS, where syncpoint processing is turned on by default.

NCIUSERID
Associates security information (UserId) with the message(s) being sent. The receiving application may use the password to authenticate the user.

Behaviour for protocol TCP/IP: The NCI communication manager verifies the userid and password if it is requested by the configuration parameter *SECURITY-LEVEL*.
For a description on how to configure the NCI communication manager refer to manual: *NCI Installation and Customization Guide*.

Behaviour for protocol MQSeries: In an MQseries client environment channel security exits provided by NCI can be used to implement application-level security (refer to manual: *NCI Installation and Customization Guide* for more details). In this case userid and password specified by the NCI application will be used for authentication at the MQSeries server site.

The receiver of a message can access the security information by the function call *NCIGetUserId*.

If no security information (UserId) will be provided by the application or via NCI sideinformation, the security context of the current process will be used (OS/390: RACF UserId, Unix: Effective UserId, Windows NT: Login UserId).

NCIGROUPID
Note: This function is not supported with protocol MQSeries.

Associates security information (GroupId) with the message(s) being sent. The receiving application may use the password to authenticate the user.

Behaviour for protocol TCP/IP: The NCI communication manager verifies the userid and password if it is requested by the configuration parameter *SECURITY-LEVEL*.
For a description on how to configure the NCI communication manager refer to manual: *NCI Installation and Customization Guide*.

Behaviour for protocol MQSeries: In an MQseries client environment channel security exits provided by NCI can be used to implement application-level security (refer to manual: *NCI Installation and Customization Guide* for more details). In this case userid and password specified by the NCI application will be used for authentication at the MQSeries server site.

The receiver of a message can access the security information by the function call *NCIGetGroupId*.

Special behaviour for operating system OS/390:
If no security information (GroupId) is provided by the application nor via NCI sideinformation, the security context (RACF GroupId) of the current process (OS/390 address space) will be used.

NCIPASSWORD
Associates security information (Password) with the message(s) being sent. The receiving application may use the password to authenticate the user.

Behaviour for protocol TCP/IP: The NCI communication manager verifies the userid and password if it is requested by the configuration parameter *SECURITY-LEVEL*.
For a description on how to configure the NCI communication manager refer to manual: *NCI Installation and Customization Guide*.

Behaviour for protocol MQSeries: In an MQseries client environment channel security exits provided by NCI can be used to implement application-level security (refer to manual: *NCI Installation and Customization Guide* for more details). In this case userid and password specified by the NCI application will be used for authentication at the MQSeries server site.

NCINEWPASSWORD
Associates security information (NewPassword) with the message(s) being sent.

Behaviour for protocol TCP/IP:
If the receiving application is running under control of an OS/390 NCI Communication Manager, the RACF password of the associated UserId will be changed if the follwoing conditions are met:

- UserId set by parameter *NCIUSERID* or NCI sideinformation is a valid RACF UserId.
- Password set by parameter *NCIPASSWORD* or NCI sideinformation is valid.
- NewPassword set by parameter *NCINEWPASSWORD* matches the RACF password rules.

NCIRETRYNUMBER
In case of a connection error, the retry number value specifies how many times a retry to establish the connection will be done.
A value of 0 means no retry. A value of 255 means unlimited number of retries.

The time interval between each retry must be specified in the parameter *NCIRETRYTIME* or NCI sideinformation.

Connection errors for protocol TCP/IP can occur if:

- the requested server has not been started
- the requested host is not up
- the network is unavailable

Connection errors for protocol MQSeries can occur if:

- the local queue manager is not active
- the requested host is not up (MQ client connection only)
- the network is unavailable (MQ client connection only)

NCIRETRYTIME
Specify a retry time value in seconds. If a connection error occurs, the *RetryTime* value will be used as wait time until the next attempt to establish a connection is started.

The number of retries must be specified by the parameter NCIRETRYNUMBER or via NCI sideinformation.

Connection errors for protocol TCP/IP can occur if:

- the requested server has not been started
- the requested host is not up
- the network is unavailable

Connection errors for protocol MQSeries can occur if:

- the local queue manager is not active
- the requested host is not up (MQ client connection only)
- the network is unavailable (MQ client connection only)

NCITIMEOUT
Specify a timeout value in seconds. After the timeout value has expired, the application gets control even if the request cannot be satisfied. If a timeout has occured, the application will be notified by a return code of **NCI_RC_INFO** (4) and a reason code of **NCI_RCC_TIMEOUT** (160).

<u>Behaviour for protocol MQSeries:</u>

Value	**Description**
0	<u>NCIPutGet:</u> The application is blocked until a reply message will be received or until a connection error occurs. <u>NCIGet:</u> Return immediately if no suitable message can be recieved. The application does not wait for messages.
1-86400	<u>NCIPutGet:</u> The maximum time, expressed in seconds, to wait for a reply message to arrive. A value of 86400 means unlimited wait time (same as 0). <u>NCIGet:</u> The maximum time, expressed in seconds, that the MQGET call waits for a suitable message to arrive. A value of 86400 means unlimited wait time.

Note: For function <u>NCIPut</u> the timeout value is not supported.

<u>Behaviour for protocol TCP/IP:</u>

Value	**Description**
0	<u>NCIPutGet:</u> The application is blocked until a reply message will arrive or until a connection error will occur.
1-86400	<u>NCIPutGet:</u> The maximum time, expressed in seconds, to wait for a reply message to arrive. A value of 86400 means unlimited wait time (same as 0).

3.2.12.4 Return Codes

Code	**Description**
0	Successful completion

12	Error (NCI handle is missing or invalid)
16	Error (unrecoverable error)

4 ncisapi0 - Setting Line Command Parameters

SAP doesn't support setting additional line command parameters for a RCF server program on the destination definition. To specify line command parameters for the NCI RFC server program it is necessary to call the RFC server program indirectly. This can be done by specifying a shell script as SAP destination and then invoking the RFC server program within the shell script with the RFC server start parameters plus the additional program specific parameters.
Below a sample shell script for various Unix systems is listed. The shell script sets some environment variables and then invokes the NCI RFC server program with additional line command parameters.

```
#  ncisapi0.sh
#
#  $Id: ncisapi0.sh,v 1.2 2003/06/12 09:16:40 saz1212 Exp $
#
#  Use this script to indirectly call the NCI SAP interface program
#  ncisapi0.
#  This is useful
#    o if the NCI components were not installed into system
#      directories and thus the shared library search path must be defined by a
#      platform specific environment variable
#    o if you want to pass some additional parameters to ncisapi0
#      - to activate logging and tracing ncisapi0 have to be called with the
#        initial parameters LOGPATH=... TRACEOPT=...
#      - to overwrite the default name space of /DBN/ ncisapi0 have to be called with
#        the initial parameter NAMESPACE=...
#
#  Note: $NCIHOME must point to the directory where the NCI
#        components have been installed.

# Set environment variable for shared library search path on HP-UX
export SHLIB_PATH=$NCIHOME/lib

# Set environment variable for shared library search path on Sun Solaris
# export LD_LIBRARY_PATH=$NCIHOME/lib

# Set environment variable for shared library search path on IBM AIX
# export LIBPATH=$NCIHOME/lib

# In the following three different invocation methods of the NCI SAP RFC server are
listed
# Note: Only one of the methods can be used at a time. The other 2 must be commented.

# 1) Invoke NCI RFC server program without Logging and Tracing and with the default
name space
exec $NCIHOME/bin/ncisapi0 $*

# 2) Invoke NCI RFC server program with traceing and logging. The log file is written
in the
# /tmp directory
#exec $NCIHOME/bin/ncisapi0 $* LOGPATH=/tmp TRACEOPT=ALL

# 3) Invoke NCI RFC server program with the name space Z_
#    eg. function NCI_OPEN can be called from ABAP with the name Z_NCI_OPEN. This
also valid for
#    all other functions provided by the NCI SAP RFC Server
#exec $NCIHOME/bin/ncisapi0 $* NAMESPACE=Z_
```

Figure 4-1: ncisapi0 Shell Script Example

5 Logging and Tracing

The NCI SAP RFC Server Interface includes logging and tracing capabilities that are, by default, deactivated. However in case of an error it is useful to active logging or tracing to analyze the error.
To activate logging and tracing the NCI RFC server program ncisapi0 have to be invoked with the line command parameters listed below. How to specify the line command parameters please refer to chapter "4 ncisapi0 - Setting Line Command Parameters".

5.1 Activating Logging

To activate logging the NCI RFC server must be invoked with the line command parameter

LOGPATH=*logpath*

were *logpath* specifies the path were the logfile will be written to.
The name of the logfile is internally generated and has the following format

NCISAPI0-log.*ccyymmdd-hhmmss-pid*

5.2 Activating Tracing

To activate tracing, logging must be also activated. Because the trace messages are written to the message log. Additionally the line command parameter

TRACEOPT=*trace-options*

has to be specified. The trace-options define the level of tracing. Currently the following trace-options are supported.

Option	Description
FUNC	Activates the function trace. Writes a trace message for each invoked function
DATA	Traces the data passed from the ABAP to the NCI RFC server and vice versa.
API	Activates tracing of the NCI API trace. This option has the same effect as setting TraceOpt(ALL) in the NCI side information file.

5.2.1 Setting the line command parameters

Please refer to chapter "4 ncisapi0 - Setting Line Command Parameters"

6 ABAP Sample

```
 FUNCTION Z_NCIPING.

 *"----------------------------------------------------------------------
 *"*"Local interface:
 *"----------------------------------------------------------------------
 data: handle      like ZNCISRV-handle,
        rc         like ZNCISRV-returncode,
        rs         like ZNCISRV-reasoncode,
        inlength   like ZNCISRV-inlength,
        inbuffer   like ZNCISRV-inbuffer.


 *Preprare NCI Server
  call function '/DBN/NCI_OPEN' destination 'NCI'
       importing
            ncihandle     = handle
            ncireturncode = rc
            ncireasoncode = rs
       exceptions
            others        = 1.

  write: / sy-subrc.
  write: / 'Handle after open: ', handle.
  write: / 'RC     after open: ', rc.
  write: / 'RS     after open: ', rs.

 *Set Addressing information
 * - use protocol MQSeries
 * - connect to Queue Manager MQX1 and
 *   Queue SYSTEM.DEFAULT.LOCAL.QUEUE
  call function '/DBN/NCI_SET_ADDR_INFO' destination 'NCI'
       exporting
            ncihandle       = handle
            nciaddrtype     = 'MQ'
            nciprimaddrinfo = 'MQX1'
            ncisecaddrinfo  = 'SYSTEM.DEFAULT.LOCAL.QUEUE'
       importing
            ncireturncode = rc
            ncireasoncode = rs
       exceptions
            others        = 1.

  write: / sy-subrc.
  write: / 'RC     after set_addr_info: ', rc.
  write: / 'RS     after set_addr_info: ', rs.

*Write message to queue and wait for reply
* on export NCIINLENGTH contains the max. length of the reply message
* on import NCIINLENGTH contains the real length of the received
* message and NCIINBUFFER contains the message
call function '/DBN/NCI_PUT_GET' destination 'NCI'
      exporting
           ncihandle       = handle
           nciinlength     = 30000
           ncioutlength    = 22
           ncioutbuffer    = 'Hallo das ist ein Test'
      importing
           ncireturncode = rc
           ncireasoncode = rs
           nciinbuffer   = inbuffer
           nciinlength   = inlength
      exceptions
           others        = 1.

 write: / sy-subrc.
 write: / 'RC          after put_get: ', rc.
 write: / 'RS          after put_get: ', rs.
 write at /(inlength)  inbuffer.

*Destroy NCI environment
 call function '/DBN/NCI_CLOSE' destination 'NCI'
      exporting
           ncihandle       = handle
      importing
           ncireturncode = rc
           ncireasoncode = rs
      exceptions
           others        = 1.

ENDFUNCTION.
```

Figure 6-1: ABAP Sample

Index

Backpage

T-Systems Enterprise Services GmbH
Computing Services & Solutions (CSS)
System Products & Automation (MSY-PA)
Fasanenweg 9, 70771 Leinfelden-Echterdingen, Germany

Phone : +49 89/1011-4687
Fax. : +49 711/972-91622
E-mail : cc.middleware@t-systems.com
Internet : http://www.t-systems-systemproducts.com

```
}
```