Middleware Suite - Application Integration

NCI - Network Computing Interface

Installation and Customization for Distributed Systems (Unix, Windows)
Installation 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

Installation and Customization Guide on Distributed Systems

Version / Revision	Date	Revision
This edition relates to NCI version NCI 3.1 • PNCI310/QZ05046 on z/OS • PNCI310/REL1003 on Unix/Windows	01/02/2006 16:59:00	23

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

Acknowledgment:

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)
This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)
This product includes software written by Tim Hudson (tjh@cryptsoft.com)
This product includes software developed by the Apache Software Foundation (http://www.apache.org/)

Table of Contents

Table of Contents **5**

List of Figures **7**

List of Tables **10**

1 Installation – Unix & Windows **13**

1.1 Installation AIX 13
1.1.1 Delivery Information and directory structure 13
1.1.2 Installation Steps 14
1.1.2.1 Installing the legacy compressed tar archive 14
1.1.2.2 Installing the bff archive 15
1.1.3 Compiling, linking and runtime 15
1.1.4 Installation Verification 16

1.2 Installation Sun Solaris 19
1.2.1 Delivery Information and directory structure 19
1.2.2 Installation Steps 20
1.2.2.1 Install NCI to system directories 20
1.2.2.2 Installing the pkg archive 21
1.2.3 Compiling, linking and runtime 21
1.2.4 Installation Verification 22

1.3 Installation HPUX 25
1.3.1 Delivery Information and directory structure 25
1.3.2 Installation Steps 26
1.3.2.1 Installing the legacy compressed tar archive 26
1.3.2.2 Installing the depot archive 26
1.3.3 Compiling, linking and runtime 27
1.3.4 Installation Verification 28

1.4 Installation LINUX 31
1.4.1 Delivery Information and directory structure 31
1.4.2 Installation Steps 32
1.4.2.1 Installing the legacy compressed tar archive 32
1.4.2.2 Installing the rpm archive 33
1.4.3 Compiling, linking and runtime 33
1.4.4 Installation Verification 34

1.5 Installation Windows 9x/2000/XP 37
1.5.1 Delivery Information and directory structure 37
1.5.2 Installation Steps 38
1.5.2.1 Installing the zipped archive 38
1.5.2.2 Installing the zipped InstallShield file 38
1.5.3 Compile and Link options 38
1.5.4 Installation Verification 38

2 Installation – Old Operating Systems **43**

2.1 Installation Digital Unix 43
2.1.1 Delivery Information and directory structure 43
2.1.2 Installation Steps 43
2.1.3 Compiling, linking and runtime 44
2.1.4 Installation Verification 45

2.2 Installation SCO 49
2.2.1 Delivery Information and directory structure 49
2.2.2 Installation Steps 49
2.2.3 Compiling, linking and runtime 50
2.2.4 Installation Verification 51

2.3 Installation Tandem/NSK 54
2.3.1 Delivery Information and directory structure 54
2.3.2 Installation Steps 54
2.3.3 Installation Verification 54

2.4 Installation OS/2 55
2.4.1 Delivery Information and directory structure 55
2.4.2 Installation Steps 56
2.4.3 Compile and Link options 56
2.4.4 Installation Verification 56

3 Customization - Unix & Windows 59

3.1 NCI Communication Manager for Unix 59
3.1.1 General Introduction 59
3.1.2 Descrpition of the NCI Communication Manager interface 59
3.1.3 Configuration File 61
3.1.3.1 Global Parameters 61
3.1.3.2 TP Specific Parameters 64
3.1.3.3 Sample Configuration File 67

3.2 NCI Communication Manager for Windows (Win32) 69
3.2.1 General Introduction 69
3.2.2 Descrpition of the NCI Communication Manager interface 69
3.2.2.1 System Service 71
3.2.3 Configuration File 73
3.2.3.1 Global Parameters 73
3.2.3.2 TP Specific Parameters 76
3.2.3.3 Sample Configuration File 79

4 Sideinformation 81

4.1 Parameter description 82

Index 105

List of Figures

Figure 1-1: Compiler options 15

Figure 1-2: Linker options 15

Figure 1-3: Runtime options 15

Figure 1-4: Compiler options 21

Figure 1-5: Linker options 21

Figure 1-6: Runtime options 21

Figure 1-7: Compiler options 27

Figure 1-8: Linker options 27

Figure 1-9: Linker options 27

Figure 1-10: Runtime options 28

Figure 1-11: Compiler options 33

Figure 1-12: Linker options 33

Figure 1-13: Runtime options 34

Figure 2-1: Compiler options 44

Figure 2-2: Linker options 44

Figure 2-3: Runtime options 44

Figure 2-4: Compiler options 50

Figure 2-5: Linker options 50

Figure 2-6: Runtime options 50

Figure 3-1: NCI Communication Manager for Unix 59

Figure 3-2: Sample NCI Configuration File for UNIX 68

Figure 3-3: NCI Communication Manager for Windows (Win32) 69

Figure 3-4: Sample NCI Configuration File for Win32 79

Figure 4-1: Example NCI sideinformation file 81

Figure 4-2: z/OS ISPF Dialog 82

Figure 4-3: Accounting Information 82

Figure 4-4: Addressing Type 83

Figure 4-5: Application Identifier 84

Figure 4-6: Cryptkey 84
Figure 4-7: Data Compression 85
Figure 4-8: Data Conversion 85
Figure 4-9: Data Encryption 86
Figure 4-10: Error Message File 86
Figure 4-11: Error Message Processing Options 87
Figure 4-12: Group Identifier 87
Figure 4-13: Keep Connection 88
Figure 4-14: Local Code Page 88
Figure 4-15: MQChillib Definition 90
Figure 4-16: MQChaltab Definition 90
Figure 4-17: MQ Correlation Identifier 90
Figure 4-18: MQ Message Expiration Period 91
Figure 4-19: MQ Message Identifier 91
Figure 4-20: MQ Message Priority 91
Figure 4-21: MQServer Definition 92
Figure 4-22: New Password 92
Figure 4-23: Password 93
Figure 4-24: Primary Addressing Information 93
Figure 4-25: Retry Number 93
Figure 4-26: Retry Time 94
Figure 4-27. Secondary Addressing Information 95
Figure 4-28: Service Identifier 96
Figure 4-29: SetParam 96
Figure 4-30: Sideinformation Application Data 98
Figure 4-31: Symbolic Name 98
Figure 4-32: Syncpoint Option 99
Figure 4-33: Timeout 100

Figure 4-34: Trace File 101

Figure 4-35: Trace Options 102

Figure 4-36: User Identifier 103

List of Tables

Table 1-1: NCI Directory Structure 13

Table 1-2: NCI Samples 14

Table 3-1: NCIXCM Command Line Parameters 61

Table 3-2: NCI Service Manager for Windows Options 72

1 Installation – Unix & Windows

1.1 Installation AIX

1.1.1 Delivery Information and directory structure

Since Version 3.1 NCI for AIX is shipped in different packages and formats. One package format is the legacy COMPRESSed TAR archive file, the other package format is the AIX platform specific archive bff format.

In general the name contains the NCI version prefix, e.g. pnci311 or nci-3.1.1 or similar, the NCI release number, e.g. REL1003 or 1003 or similar, the minimal operating system level, e.g. aix-4.3 or aix-5.1, and possibly the processor architecture, e.g. powerpc. The resulting complete name depends on the specific package. The 64bit version is supported since aix-5.1 or higher, it is already included in the related bff archive, whereas in case of the compressed tar archive an explicit 64bit package exists, which is tagged accordingly, e.g. contains a _64 tag.

Note: The Release number could be higher than REL1003.

The following directories will be created in the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

Table 1-1: NCI Directory Structure

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables
ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ". ./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

Table 1-2: NCI Samples

1.1.2 Installation Steps

1.1.2.1 Installing the legacy compressed tar archive

After transferring the archive pnci311.REL1003.aix-5.1.tar.Z to the target platform the *compress* and *tar* command must be used to extract the archive:

```
compress -c -d pnci311.REL1003.aix-5.1.tar.Z | tar -xvf –
```

Install NCI to system directories

If NCI should be available for all users or projects, it's recommended to copy all NCI elements to the system directories.

- Copy NCI shared library files *$NCIHOME/lib/** to */usr/lib/*
- Copy NCI executables *$NCIHOME/bin/**to */usr/bin/*
- Copy NCI header files *$NCIHOME/include/**to */usr/include/*

Install NCI to a private directory

If NCI should be available to only specific users or projects, it's recommended to install it to a private directory.
If the NCI components like header files and shared libraries are installed in a private directory special care must be taken while compiling, linking and running applications using NCI.

1.1.2.2 Installing the bff archive

Since NCI 3.1.0, for Unix platforms NCI is also shipped in the platform specific archive format. In this format, NCI is always installed to /opt/nci. It is not possible to install several versions at one time. In order to perform this installation *root* permissions are necessary.

For AIX platforms this is the bff format . After transferring the archive to the target platform, the sotware must be installed using the system administration tool *smitty* to install the software. Launch *smitty* and follow the menu structure to software maintenance. It is possible to use the command line */usr/lib/instl/sm_inst* . For detailed information see IBM AIX Administration documentation.

1.1.3 Compiling, linking and runtime

Compiler options must be used to expand the include search path, telling the compiler where to find the *nci.h* header file.

```
#include "nci.h"

compiler searches the nci.h header file in the current
directory or the search may be expanded using the
-I compiler option

cc -c -I $NCIHOME/include myfile.c
```

Figure 1-1: Compiler options

Linker options must be used to let the linker know where to find the NCI shared library.

```
ld -L $NCIHOME/lib -lnci myobj.o

Tells the linker that the NCI shared library must be
searched in the directory $NCIHOME/lib.
```

Figure 1-2: Linker options

To execute the NCI sample applications the environment variable *PATH* must include the NCI *$NCIHOME/bin* directory (export PATH=$NCIHOME/bin:$PATH).
During runtime NCI applications have to access the NCI shared library. Therefore it is necessary to set a system environment variable to tell the loader where to find the NCI shared library.

```
Search order for shared libraries:
               1. LIBPATH environment variable
               2. directory where the shared lib was found while
                  linking the application.
               3. system libraries. eg. /usr/lib
```

Figure 1-3: Runtime options

1.1.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

   ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

   By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

   If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI310 $Name:REL1003$
Jun  6 2000 16:06:37 2095  starting...
NCI7276I NCIXCM   : Use shared memory segment for key 5800/0x000016a8.
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
at TCP/IP port (3450).
```

   Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

   If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

   The NCI Communication Manager can be stopped by sending the TERM signal to the Communication Manager process

   Obtain the process ID and send the TERM signal.

```
> ps

     PID    TTY  TIME CMD
     16796  pts/0  0:00 ncixsvd
     17056  pts/0  0:00 ps
     17818  pts/0  0:00 ksh

> kill 16796
```

While NCI Communication Manager stops the following messages will be written:

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Unix"on page 59.

1.2 Installation Sun Solaris

1.2.1 Delivery Information and directory structure

Since Version 3.1 NCI for Sun Solaris is shipped in different packages and formats. One package format is the legacy COMPRESSed TAR archive file, the other package format is the Sun Solaris platform specific archive pkg format.

In general the name contains the NCI version prefix, e.g. pnci311 or nci-3.1.1 or similar, the NCI release number, e.g. REL1003 or 1003 or similar, the minimal operating system level, e.g. sun-5.6 or sun-5.8, and possibly the processor architecture, e.g. sparc. The resulting complete name depends on the specific package. The 64bit version is supported since sun-5.8 or higher, it is already included in the related pkg archive, whereas in case of the compressed tar archive an explicit 64bit package exists, which is tagged accordingly, e.g. contains a _64 tag.

Note: The Release number could be higher than REL1003.

The following directories will be extracted to the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager

File	Description
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables
ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ". ./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

1.2.2 Installation Steps

1.2.2.1 Install NCI to system directories

After transferring the archive pnci311.REL1003.sun-5.8.tar.Z to the target platform the *compress* and *tar* command must be used to extract the archive:

```
compress -c -d pnci311.REL1003.sun-5.8.tar.Z | tar -xvf –
```

Install NCI to system directories

If NCI should be available for all users or projects, it's recommended to copy all NCI elements to the system directories.

- Copy NCI shared library files *$NCIHOME/lib/** to */usr/lib/*
- Copy NCI executables *$NCIHOME/bin/** to */usr/bin/*
- Copy NCI header files *$NCIHOME/include/** to */usr/include/*

Install NCI to a private directory

If NCI should be available to only specific users or projects, it's recommended to install it to a private directory.
If the NCI components like header files and shared libraries are installed in a private directory special care must be taken while compiling, linking and running applications using NCI.

1.2.2.2 Installing the pkg archive

Since NCI 3.1.0, for Unix platforms NCI is also shipped in the platform specific archive format. In this format, NCI is always installed to /opt/nci. It is not possible to install several versions at one time. In order to perform this installation *root* permissions are necessary.

For Sun Solaris platforms this is the pkg format . After transferring the archive to the target platform, the sotware must be installed using the system administration tool *pkgadd* to install the software. This can be done by typing:

pkgadd -d [directory of package location]

For detailed information see Sun Solaris Administration documentation.

1.2.3 Compiling, linking and runtime

Compiler options must be used to expand the include search path, telling the compiler where to find the *nci.h* header file.

```
#include "nci.h"

compiler searches the nci.h header file in the current
directory or the search may be expanded using the
-I compiler option

cc -c -I $NCIHOME/include myfile.c
```

Figure 1-4: Compiler options

Linker options must be used to let the linker know where to find the NCI shared library.

```
ld -L $NCIHOME/lib -lnci myobj.o

Tells the linker that the NCI shared library must be
searched in the directory $NCIHOME/lib.
```

Figure 1-5: Linker options

To execute the NCI sample applications the environment variable *PATH* must include the NCI *$NCIHOME/bin* directory (export PATH=$NCIHOME/bin:$PATH).
During runtime NCI applications have to access the NCI shared library. Therefore it is necessary to set a system environment variable to tell the loader where to find the NCI shared library.

```
Search order for shared libraries:
                1. LD_LIBRARY_PATH environment variable
                2. system library search path
```

Figure 1-6: Runtime options

1.2.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (xx bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

   ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

   By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

   If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI310 $Name:REL1003$
Jun  6 2000 16:06:37 2095  starting...
NCI7276I NCIXCM   : Use shared memory segment for key 5800/0x000016a8.
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
at TCP/IP port (3450).
```

   Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

   If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

   The NCI Communication Manager can be stopped by sending the TERM signal to the Communication Manager process

   Obtain the process ID and send the TERM signal.

```
> ps

     PID    TTY  TIME CMD
     16796  pts/0  0:00 ncixsvd
     17056  pts/0  0:00 ps
     17818  pts/0  0:00 ksh

> kill 16796
```

While NCI Communication Manager stops the following messages will be written:

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Unix"on page 59.

1.3 Installation HPUX

1.3.1 Delivery Information and directory structure

Since Version 3.1 NCI for HPUX is shipped in different packages and formats. One package format is the legacy COMPRESSed TAR archive file, the other package format is the HPUX platform specific archive depot format.

In general the name contains the NCI version prefix, e.g. pnci311 or nci-3.1.1 or similar, the NCI release number, e.g. REL1003 or 1003 or similar, the minimal operating system level, e.g. hpux-11.00, and possibly the processor architecture, e.g. hppa. The resulting complete name depends on the specific package. The 64bit version is supported since hpux-11.00 or higher, it is already included in the related depot archive, whereas in case of the compressed tar archive an explicit 64bit package exists, which is tagged accordingly, e.g. contains a _64 tag.

Note: The Release number could be higher than REL1003.

The following directories will be created in the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables

ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ". ./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

1.3.2 Installation Steps

1.3.2.1 Installing the legacy compressed tar archive

After transferring the archive pnci311.REL1003.hpux-11.00.tar.Z to the target platform the *compress* and *tar* command must be used to extract the archive:

```
compress -c -d pnci311.REL1003.hpux-11.00.tar.Z | tar -xvf –
```

Install NCI to system directories

If NCI should be available for all users or projects, it's recommended to copy all NCI elements to the system directories.

- Copy NCI shared library files *$NCIHOME/lib/** to */usr/lib/*
- Copy NCI executables *$NCIHOME/bin/** to */usr/bin/*
- Copy NCI header files $NCIHOME/include/* to /usr/include/

Install NCI to a private directory

If NCI should be available to only specific users or projects, it's recommended to install it to a private directory.
If the NCI components like header files and shared libraries are installed in a private directory special care must be taken while compiling, linking and running applications using NCI.

1.3.2.2 Installing the depot archive

Since NCI 3.1.0, for Unix platforms NCI is also shipped in the platform specific archive format. In this format, NCI is always installed to /opt/nci. It is not possible to install several versions at one time. In order to perform this installation *root* permissions are necessary.

For HPUX platforms this is the depot format . After transferring the archive to the target platform, the sotware must be installed using the system administration tool *swinstall* to install the software. This can be done by typing:

swinstall -s [fully qualified download directory and filename]

For detailed information see HP HPUX Administration documentation.

1.3.3 Compiling, linking and runtime

Compiler options must be used to expand the include search path, telling the compiler where to find the *nci.h* header file.

```
#include "nci.h"

compiler searches the nci.h header file in the current
directory or the search may be expanded using the
-I compiler option

cc -c -I $NCIHOME/include myfile.c
```

Figure 1-7: Compiler options

Linker options must be used to let the linker know where to find the NCI shared library.

```
ld -L $NCIHOME/lib -lnci myobj.o

Tells the linker that the NCI shared library must be
searched in the directory $NCIHOME/lib.
```

Figure 1-8: Linker options

Note: When using a multithreaded environment, e.g. that is often the case in oracle SQL applications, and the MQ interface so it is required to use the MQSeries multithread libraries. This can be easily done by link with the apropriate NCI library: libnci_r.sl.

```
ld -L $NCIHOME/lib -lnci_r myobj.o
```

Figure 1-9: Linker options

To execute the NCI sample applications the environment variable *PATH* must include the NCI *$NCIHOME/bin* directory (export PATH=$NCIHOME/bin:$PATH).
During runtime NCI applications have to access the NCI shared library. Therefore it is necessary to set a system environment variable to tell the loader where to find the NCI shared library.

```
Search order for shared libraries:
               1. SHLIB_PATH environment variable (see Note below)
               2. system library search path

Note: Linkage editor option: -Wl,+s must be used to tell the
      shared library loader to use environment variable: SHLIB_PATH at run-time, to
search
      for shared library objects.
```

Figure 1-10: Runtime options

1.3.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (xx bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

   ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

   By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

   If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI310 $Name:REL1003$
Jun  6 2000 16:06:37 2095  starting...
NCI7276I NCIXCM   : Use shared memory segment for key 5800/0x000016a8.
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
at TCP/IP port (3450).
```

   Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

   If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

   The NCI Communication Manager can be stopped by sending the TERM signal to the Communication Manager process

   Obtain the process ID and send the TERM signal.

```
> ps

     PID    TTY  TIME CMD
     16796  pts/0  0:00 ncixsvd
     17056  pts/0  0:00 ps
     17818  pts/0  0:00 ksh

> kill 16796
```

   While NCI Communication Manager stops the following messages will be written:

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Unix"on page 59.

1.4 Installation LINUX

1.4.1 Delivery Information and directory structure

Since Version 3.1 NCI for Linux is shipped in different packages and formats. One package format is the legacy COMPRESSed TAR archive file, the other package format is the Linux platform specific archive rpm format. It is used for both SuSE and RedHat installations.

In general the name contains the NCI version prefix, e.g. pnci311 or nci-3.1.1 or similar, the NCI release number, e.g. REL1003 or 1003 or similar, the minimal operating system level, e.g. linux-2.4 or linux-2.6, and possibly the processor architecture, e.g. i386 or s390. The resulting complete name depends on the specific package. The 64bit version is not yet supported.

Note: The Release number could be higher than REL1003.

The following directories will be extracted to the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables
ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ". ./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

1.4.2 Installation Steps

1.4.2.1 Installing the legacy compressed tar archive

After transferring the archive pnci311.REL1003.linux-2.4.tar.Z to the target platform the *compress* and *tar* command must be used to extract the archive:

```
compress -c -d pnci311.REL1003.linux-2.4.tar.Z | tar -xvf –
```

Install NCI to system directories

If NCI should be available for all users or projects, it's recommended to copy all NCI elements to the system directories.

- Copy NCI shared library files *$NCIHOME/lib/** to */usr/lib/*
- Copy NCI executables *$NCIHOME/bin/** to */usr/bin/*
- Copy NCI header files *$NCIHOME/include/** to */usr/include/*

Install NCI to a private directory

If NCI should be available to only specific users or projects, it's recommended to install it to a private directory.
If the NCI components like header files and shared libraries are installed in a private directory special care must be taken while compiling, linking and running applications using NCI.

1.4.2.2 Installing the rpm archive

Since NCI 3.1.0, for Unix platforms NCI is also shipped in the platform specific archive format. In this format, NCI is always installed to /opt/nci. It is not possible to install several versions at one time. In order to perform this installation *root* permissions are necessary.

For Linux platforms this is the rpm format . After transferring the archive to the target platform, the sotware must be installed using the system administration tool *rpm* to install the software. This can be done by typing:

rpm -ihv –nodeps [archive filename]

For detailed information see Linux manpages oder RedHat rpm documentation.

1.4.3 Compiling, linking and runtime

Compiler options must be used to expand the include search path, telling the compiler where to find the *nci.h* header file.

```
#include "nci.h"

compiler searches the nci.h header file in the current
directory or the search may be expanded using the
-I compiler option

cc -c -I $NCIHOME/include myfile.c
```

Figure 1-11: Compiler options

Linker options must be used to let the linker know where to find the NCI shared library.

```
ld -L $NCIHOME/lib -lnci myobj.o

Tells the linker that the NCI shared library must be
searched in the directory $NCIHOME/lib.
```

Figure 1-12: Linker options

To execute the NCI sample applications the environment variable *PATH* must include the NCI *$NCIHOME/bin* directory (export PATH=$NCIHOME/bin:$PATH).
During runtime NCI applications have to access the NCI shared library. Therefore it is necessary to set a system environment variable to tell the loader where to find the NCI shared library.

```
Search order for shared libraries:
               1. LD_LIBRARY_PATH environment variable
               2. Directories defined in the file /etc/ld.so.conf.
                  Changes in /etc/ld.so.conf can be activated with
                  the command: ldconfig
```

Figure 1-13: Runtime options

1.4.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (xx bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

   ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

   By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

   If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI310 $Name:REL1003$
Jun  6 2000 16:06:37 2095  starting...
NCI7276I NCIXCM   : Use shared memory segment for key 5800/0x000016a8.
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
at TCP/IP port (3450).
```

   Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

   If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

   The NCI Communication Manager can be stopped by sending the TERM signal to the Communication Manager process

   Obtain the process ID and send the TERM signal.

```
> ps

     PID    TTY  TIME CMD
     16796  pts/0  0:00 ncixsvd
     17056  pts/0  0:00 ps
     17818  pts/0  0:00 ksh

> kill 16796
```

While NCI Communication Manager stops the following messages will be written:

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Unix" on page 59

1.5 Installation Windows 9x/2000/XP

1.5.1 Delivery Information and directory structure

NCI for Windows is shipped as a zipped InstallShield file and as a zipped archive file.

In general the name contains the NCI version prefix, e.g. pnci311 or nci-3.1.1 or similar, the NCI release number, e.g. REL1003 or 1003 or similar, and the minimal operating system level, e.g. win. The resulting complete name depends on the specific package. The 64bit version is not yet supported.

Note: The Release number could be higher than REL1003.

The following directories will be extracted to the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables
ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ".

	./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

1.5.2 Installation Steps

1.5.2.1 Installing the zipped archive

Unzip the pnci311.REL1003.win.zip in a directory.

If NCI should be available for all applications, it's recommended to set the environment variable path to include the $NCIHOME\bin directory

- set path=%path%;$NCIHOME\bin

Alternatively you can copy the NCI dynamic load libraries to the windows system directory

- copy $NCIHOME\bin*.dll to winnt\system32\

1.5.2.2 Installing the zipped InstallShield file

Unzip the pnci311.REL1003.win.exe.zip and execute afterwards the file pnci311.REL1003.win.exe to start the installation wizard.

Optionally you can copy the NCI dynamic load libraries to the windows system directory

- (copy $NCIHOME\bin*.dll to winnt\system32\)

1.5.3 Compile and Link options

Compiler options must be used to let the compiler know where to find the nci header file (nci.h) and library file (nci.lib).

- The nci header file is located at $NCIHOME\h\nci.h
- The nci library file is located at $NCIHOME\lib\nci.lib

1.5.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (xx bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI310 $Name:REL1003$
Jun  6 2000 16:06:37 2095  starting...
NCI7276I NCIXCM   : Use shared memory segment for key 5800/0x000016a8.
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
at TCP/IP port (3450).
```

Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

The NCI Communication Manager can be stopped by terminating it with the NT TaskManager or by pressing Ctrl-C in the controling console. Therefore the Operating System provide no mechanisms to interact with a process like signals, it is not possible to shutdown the NCI Communication Manager *ncixsvd* in a soft way. The WinNT offers the Ctrl-C break or the Stop from the TaskManager. With both methods, the process will be immediately stopped and no shutdown message will be written. In the case of the NCI Service (refer to "NCI Communication Manager for Windows (Win32)"on page 69) Windows provides a communication service to the process via *Stop Service* in the *System Control - Services Panel*. So a regular shutdown of the NCI Communication Manager can be performed, and the following messages will be written to file: *ncimsg.log*.

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Windows (Win32)"on page 69.

2 Installation – Old Operating Systems

2.1 Installation Digital Unix

2.1.1 Delivery Information and directory structure

NCI for DEC is shipped as a COMPRESSed TAR archive file.

The supported platform is decOSF1 version 4.0. This platform was set to frozen state after version PNCI250 and isn't supported anymore.

The following directories will be created in the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables
ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ". ./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

2.1.2 Installation Steps

After transferring the archive nci25.PNCI250.decOSF1-4.0.tar.Z to the target platform the *compress* and *tar* command must be used to extract the archive.

compress -c -d nci25.PNCI250.decOSF1-4.0.tar.Z | tar -xvf -

Install NCI to system directories

If NCI should be available for all users or projects, it's recommended to copy all NCI elements to the system directories.

- Copy NCI shared library files *$NCIHOME/lib/** to */usr/lib/*
- Copy NCI executables *$NCIHOME/bin/** to */usr/bin/*
- Copy NCI header files *$NCIHOME/include/** to */usr/include/*

Install NCI to a private directory

If NCI should be available to only specific users or projects, it's recommended to install it to a private directory.
If the NCI components like header files and shared libraries are installed in a private directory special care must be taken while compiling, linking and running applications using NCI.

2.1.3 Compiling, linking and runtime

Compiler options must be used to expand the include search path, telling the compiler where to find the *nci.h* header file.

```
#include "nci.h"

compiler searches the nci.h header file in the current
directory or the search may be expanded using the
-I compiler option

cc -c -I $NCIHOME/include myfile.c
```

Figure 2-1: Compiler options

Linker options must be used to let the linker know where to find the NCI shared library.

```
ld -L $NCIHOME/lib -lnci myobj.o

Tells the linker that the NCI shared library must be
searched in the directory $NCIHOME/lib.
```

Figure 2-2: Linker options

To execute the NCI sample applications the environment variable *PATH* must include the NCI *$NCIHOME/bin* directory (export PATH=$NCIHOME/bin:$PATH).
During runtime NCI applications have to access the NCI shared library. Therefore it is necessary to set a system environment variable to tell the loader where to find the NCI shared library.

```
Search order for shared libraries:
               1. LD_LIBRARY_PATH environment variable
               2. system library search path
```

Figure 2-3: Runtime options

2.1.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

   ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

   By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

   If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI250 Jun  6 2000 16:06:37
starting...
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
                    at TCP/IP port (3450).
```

   Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

   If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

   The NCI Communication Manager can be stopped by sending the TERM signal to the Communication Manager process

   Obtain the process ID and send the TERM signal.

```
> ps

    PID    TTY  TIME CMD
    16796  pts/0  0:00 ncixsvd
    17056  pts/0  0:00 ps
    17818  pts/0  0:00 ksh

> kill 16796
```

While NCI Communication Manager stops the following messages will be written:

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Unix"on page 59.

2.2 Installation SCO

2.2.1 Delivery Information and directory structure

NCI for SCO is shipped as a COMPRESSed TAR archive file.

The supported platform is a intel based SCO Unix version 3.2. This platform was set to frozen state after version PNCI280 and isn't supported anymore.

The following directories will be created in the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables (sample programs and nciping/ncipong utilities)
lib	NCI shared libraries
samples	NCI samples (C samples, makefiles, config-files)
include	NCI header files

The following samples are shipped in directory *samples*:

File	Description
ncixsvd.cfg	NCI sample configuration file for NCI Communication Manager
nciside	NCI sample sideinformation file
nciside.env	NCI sample script to define sideinformation via environment variables
ncisapi0.sh	NCI sample script to indirectly call the NCI SAP interface program ncisapi0
runping.sh	NCI sample script to indirectly call the NCI echo client program nciping
runxsvd.sh	NCI sample script to indirectly call the NCI Communication Manager program ncixsvd
setenv.sh	NCI sample script to set the environment variables for the local NCI installation (must be called ". ./setenv.sh")
makefile.sample	Makefile to create NCI sample applications (nci1*.c)
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

2.2.2 Installation Steps

After transferring the archive nci28.PNCI280J.sco-3.2.tar.Z to the target platform the *compress* and *tar* command must be used to extract the archive.

```
compress -c -d nci28.PNCI280J.sco-3.2.tar.Z | tar -xvf -
```

Install NCI to system directories

If NCI should be available for all users or projects, it's recommended to copy all NCI elements to the system directories.

- Copy NCI shared library files *$NCIHOME/lib/** to */usr/lib/*
- Copy NCI executables *$NCIHOME/bin/** to */usr/bin/*
- Copy NCI header files *$NCIHOME/include/** to */usr/include/*

Install NCI to a private directory

If NCI should be available to only specific users or projects, it's recommended to install it to a private directory.
If the NCI components like header files and shared libraries are installed in a private directory special care must be taken while compiling, linking and running applications using NCI.

2.2.3 Compiling, linking and runtime

Compiler options must be used to expand the include search path, telling the compiler where to find the *nci.h* header file.

```
#include "nci.h"

compiler searches the nci.h header file in the current
directory or the search may be expanded using the
-I compiler option
cc -c -I $NCIHOME/include myfile.c
```

Figure 2-4: Compiler options

Linker options must be used to let the linker know where to find the NCI shared library.

```
ld -L $NCIHOME/lib -lnci myobj.o

Tells the linker that the NCI shared library must be
searched in the directory $NCIHOME/lib.
```

Figure 2-5: Linker options

To execute the NCI sample applications the environment variable *PATH* must include the NCI *$NCIHOME/bin* directory (export PATH=$NCIHOME/bin:$PATH).
During runtime NCI applications have to access the NCI shared library. Therefore it is necessary to set a system environment variable to tell the loader where to find the NCI shared library.

```
Search order for shared libraries:
               1. LD_LIBRARY_PATH environment variable
               2. system library search path
```

Figure 2-6: Runtime options

2.2.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Parallel Server

1. Start NCI Communication Manager by command:

   ncixsvd -f $NCIHOME/samples/ncixsvd.cfg

   By default ncixsvd searches for the configuration file ncixsvd.cfg in the current working directory. A sample configuration file (named ncixsvd.cfg) is shipped in the *samples* directory. Therefore if you start the NCI Communication Manager from the *$NCIHOME/bin* directory you must specify the location of the configuration file via option *-f*.

   If the NCI Communication Manager starts sucessful the following messages will be written to *STDOUT*.

```
NCI7240I NCIXCM   : Server pid:26634, version: PNCI280 Jun  6 2000 16:06:37
starting...
NCI7248I NCIXCM   : Server NCITEST,4668 ready to accept client requests
                    at TCP/IP port (3450).
```

   Note: IP-PORT(3450) is defined in the sample NCI configuration-file as a default value. Choose any other port if this port is not applicable for you.

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server, (defined via configuration file, parameter: IP-PORT(3450))

   If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping with option -h to get help information about supported options.
4. Stop the NCI Communication Manager

   The NCI Communication Manager can be stopped by sending the TERM signal to the Communication Manager process

   Obtain the process ID and send the TERM signal.

```
> ps

    PID    TTY  TIME CMD
    16796  pts/0  0:00 ncixsvd
    17056  pts/0  0:00 ps
    17818  pts/0  0:00 ksh

> kill 16796
```

While NCI Communication Manager stops the following messages will be written:

```
Thu Jul 24 07:47:27 1997 - NCI7242I - NCIXCM Server NCITEST shutdown
requested by TERM signal.
Thu Jul 24 07:47:27 1997 - NCI7241I - NCIXCM Server NCITEST terminated.
```

For more information how to configure the NCI Communication Manager refer to "NCI Communication Manager for Unix"on page 59.

2.3 Installation Tandem/NSK

2.3.1 Delivery Information and directory structure

NCI for Tandem/NSK is shipped as a zipped archive file.

For this platform only the MQ FileTransfer utilities are supported in version PNCI280.

The following directories will be created in the installation path (later on refered as $NCIHOME)

Directory	Description
nci	NCI executables and library (sample programs and nciping/ncipong utilities)
mqft	NCI MQFT executables
config	NCI sample configuration

2.3.2 Installation Steps

After unzip the archive nci28.PNCI280Z.tandem.zip on a windows platform, please transfer the files into the appropriate place on the Tandem.

2.3.3 Installation Verification

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries

-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options

2.4 Installation OS/2

2.4.1 Delivery Information and directory structure

NCI for OS/2 is shipped as a self extracting ZIP file.

The following directories will be extracted to the installation path (later on refered as $NCIHOME)

Directory	Description
bin	NCI executables, including dynamic load libraries
lib	NCI library files
samples	NCI samples (C samples, makefiles, config-files)
h	NCI header files

The following samples are shipped in directory *samples*:

File	Description
nciside	NCI sample sideinformation file
ncienv.bat	NCI sample script to define sideinformation via environment variables
nci1ipg.c	Get a message (protocol: TCP/IP) and send back a reply message (This application runs as a single server)
nci1ippg.c	C program: Put a message and wait for a reply message (protocol: TCPIP)
nci1mqg.c	C program: Get a message from a queue using NCI direct addressing (MQSeries)
nci1mqp.c	C program: Put a message to a queue using NCI direct addressing (MQSeries)
nci1mqpg.c	C program: Put a message to a queue and wait for a reply message using NCI direct addressing (MQSeries)
nci1sig.c	C program: Get a message using symbolic addressing
nci1sip.c	C program: Put a message using symbolic addressing
nci1sipg.c	C program: PutGet a message using symbolic addressing

2.4.2 Installation Steps

Copy ZIP file nci25.PNCI250.os2.exe to an emty directory (e.g nci) and unzip self extracting file by command:

nci25.PNCI250.os2.exe -d

Option -d is required to restore all included directories.

If NCI should be available for all applications, it's recommended to set the environment variable *path* to include the $NCIHOME\bin directory

4. set path=%path%;$NCIHOME\bin.

Alternatively you can copy the NCI dynamic load libraries to the OS2 DLL directory or another DLL-directory defined via environment variable *libpath*

5. copy $NCIHOME\bin*.dll to os2\dll\

2.4.3 Compile and Link options

Compiler options must be used to let the compiler know where to find the nci header file (nci.h) and library file (nci.lib).

- The nci header file is located at $NCIHOME\h\nci.h
- The nci library file is located at $NCIHOME\lib\nci.lib

2.4.4 Installation Verification

If NCI has been installed into a private directory, then you must set environment variables in order to specify the directories where the executables and shared libraries reside. You can do this by issuing the following command from the NCI samples subdirectory:

```
. ./setenv.sh
```

Verification Test (MQSeries)

1. Start NCI echo server (ncipong) by command:

```
ncipong -a MQ -2 queuename -w 86400
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name (e.g. SYSTEM.DEFAULT.LOCAL.QUEUE)
-w 86400	unlimited waittime (wait until a message arrives)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
ncipong   -   nciSetTimeout ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a MQ -2 queuename -y Y
```

Option	Description
-a MQ	AddressingType: MQSeries
-2 queuename	MQSeries queue name of the NCI echo server (SYSTEM.DEFAULT...)
-y Y	message requires a reply

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetPrimAddrInfo ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

Verification Test (TCP/IP) Single Server

1. Start NCI echo server (ncipong) by command:

```
ncipong -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber (e.g. 3450)

If the NCI echo server starts successful the following messages appear:

```
ncipong   -   nciOpen ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetAddrType ReturnCode: 0
ncipong   -   nciSetDataConv ReturnCode: 0
ncipong   -   nciSetSecAddrInfo ReturnCode: 0
```

2. Start NCI echo client (nciping) by command:

```
nciping -a TCPIP -2 portnumber
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-2 portnumber	TCP/IP portnumber of the NCI echo server (e.g. 3450)

If the NCI echo client completes successful the following messages appear:

```
nciping   -   nciOpen ReturnCode: 0
nciping   -   nciSetAddrType ReturnCode: 0
nciping   -   nciSetSecAddrInfo ReturnCode: 0
nciping   -   sending message: Hello World
nciping   -   received reply message (46 bytes): Response from ...: Hello World
nciping   -   nciClose ReturnCode: 0
```

3. Start nciping/ncipong with option -h to get help information about supported options.

3 Customization - Unix & Windows

3.1 NCI Communication Manager for Unix

3.1.1 General Introduction

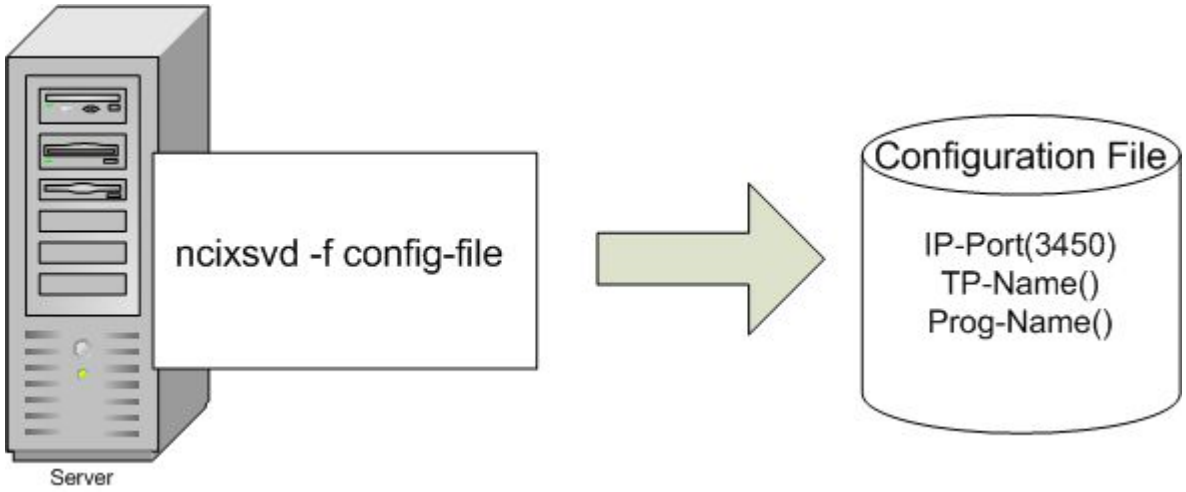


Figure 3-1: NCI Communication Manager for Unix

The NCI Communcation Manager could be started either as a command line call or by a API call. All the necessary configuration information which is needed to run the NCI Communication Manager is located in a configuration file. These configuration data could the TCP/IP Port information, which Ethernet Interface it is listening to, the TP Program information, and a lot more. To start the NCI Communcation Manager it is absolutely necessary to provide this configuration file, independent of the way it is started.

3.1.2 Descrpition of the NCI Communication Manager interface

The NCI Communication Manager accepts two kind of interfaces:

1. Program-Call interface
   NCI Communication Manager may be called from an application program as a subroutine, contained in the nci shared library.

```
#include "nci.h"

 void main(void)
{
int rc;
  /* perform some application specific processing */
  doSomething();

  /* call nci communication manger */
  rc = ncixcm(.....);

  return 0;
}
```

2. Command Line interface
   Front end to call the NCI Communication Manager from the command line. This front end only interprets the passed commandline parameters and then calls the nci Communication Manager library routine.

```
$ ncixsvd -f config-file -d
```

Parameters

NCIXCM

The library routine accepts as parameter only a pointer to a parameter string. This parameter string is a string of ncixcm parameter keywords each keyword separated by a semicolon. The following keywords are valid.

- CONFIG-FILE(name)
  Name of the nci Communication Manager configuration file. This file may contain additional NCI configuration parameters.
- all keywords which are valid for the NCI configuration file.

Example

```
#include "nci.h"

void main(void)
{
  int rc;
  char *params="CONFIG-FILE(/etc/ncicm.cfg);IP-PORT(3450);..."

  rc = ncixcm(params);

  exit
}
```

Note: If a keyword is specified in the configuration file and is additionally provided by the parameter string, the keyword from the parameter string takes precedence over the keyword in the configuration file.

Command Line interface

The command line interface accepts the following parameters.

-h	Display information about supported command line parameters.
-d	Daemonize the process. If this switch is used the process will run as a daemon.
-f configurationfile	NCI configuration file to be used by the server. If this parameter is omitted a file named <ncixsvd.cfg> in the current working directory is expected.
-s ipport	TCP/IP port number to be used by the server. This flag overwrites the IP-PORT definition within the configuration file.

-w working directory	Working directory to be used by the server daemon. If this parameter is omitted the default root directory '/' is used. This parameter will only be used if the paramter -d is also specified.

Table 3-1: NCIXCM Command Line Parameters

Example

```
$ ncixsvd -d -w /home/hugo -f hugonci.cfg
```

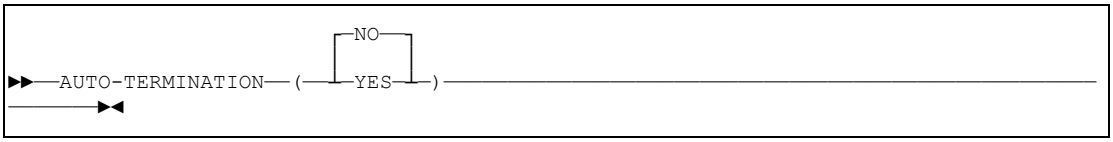
In the example listed above the Communication Manager process will be daemonized. The working directory is changed to */home/hugo* and the name of the nci configuration file is *hugonci.cfg*.

3.1.3 Configuration File

Each NCI server needs a separate configuration file. In contrast to the OS/390 platform were the nci server configuration is devided into two parts, a configuration module and a configuration file, on the UNIX and Windows NT platform all server configuration parameters are contained in the configuration file.
The configuration parameters can be divided into two categories. Global parameters used to control the characteristics of the whole server and TP specific parameters used to define a TP entry.

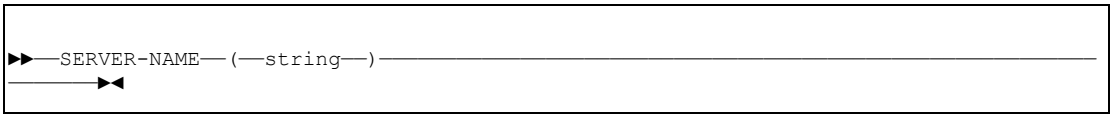
3.1.3.1 Global Parameters

AUTO-TERMINATION



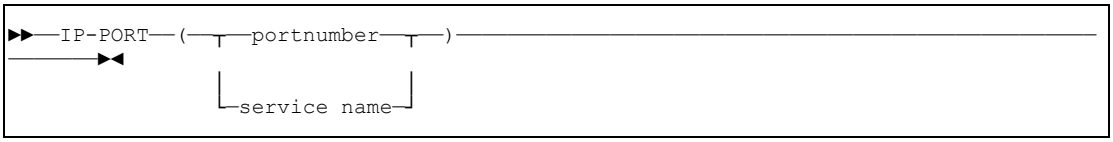
YES	Terminate the server after the last TP ended
NO	Server runs until TERM signal received.

SERVER-NAME



string	Name (1-20 characters) of the application. The string is used for NCI internal control block eye catchers and as syslog message prefix.

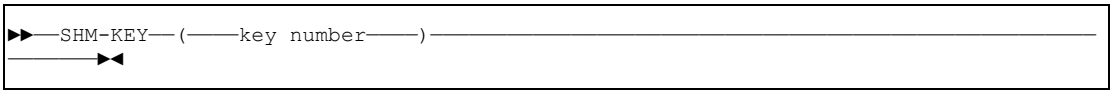
IP-PORT



portnumber IP port to be used by the NCI Communication Manager.

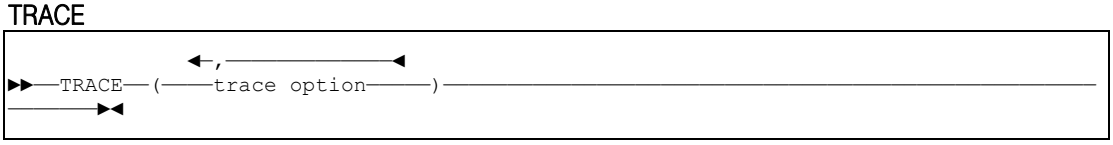
servicename Name of a service defined in the /etc/services file to be used by the NCI Communication Manager. While startup NCI Communication Manager uses the getservbyname call to obtain the port number corresponding to the defined service name.

SHM-KEY



key number Numeric digit specifying the key to be used to create a shared memory segment. Must be unique over the whole system !

TRACE



Controls the level of trace messages written to the NCI trace file. If this parameter is specified the TRACE-FILE parameter must also be specified.

Note: Due to performance impacts tracing should be only activated for error analysis.

Valid trace options are:

NONE	**Disable Trace.**
ALL	Trace all.
FUNC	Function Trace.
DATA	Tracing of data flow
OPTIONS	NCI options being used.
CB	Tracing of NCI Control Blocks
CODESET	Tracing of Character Set handling
PROT	Specific transport protocol trace
QUE	Trace of request queuing
CMC	Trace of parallel server communication
CSS	Trace of parallel server handling

API Trace of special API calls (e.g. RFC)

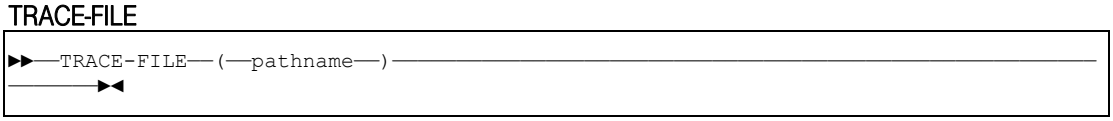
DYN Trace dynamic loading

ACCT Trace of accounting

SEC Trace of security relevant settings

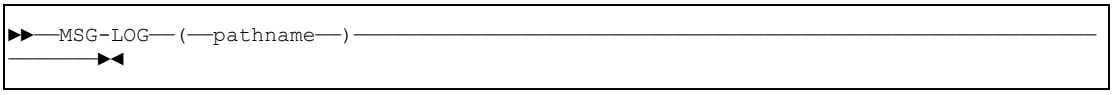
DATACNV Trace Data Conversion

TRACE-FILE



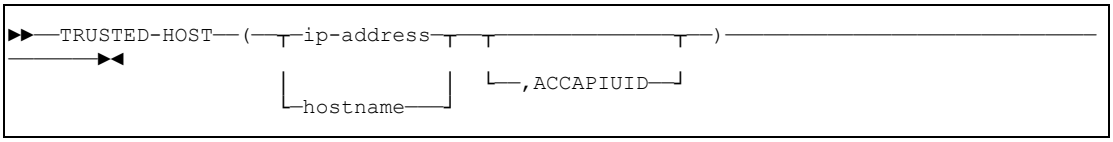
pathname Path and filename of NCI trace file.
To write trace messages to an already open file descriptor eg. stdout, the output stream may be specified as numeric digit.

MSG-LOG



pathname Path and filename of NCI message log file.
To write trace messages to an already open file descriptor eg. stdout, the output stream may be specified as numeric digit.

TRUSTED-HOST



Parameter specifies hosts allowed to access a TP defined with security PROPAGATE or CHECK only with a valid userid but without a password. For all client hosts defined as trusted, the NCI server assumes that the userid has been already verified.

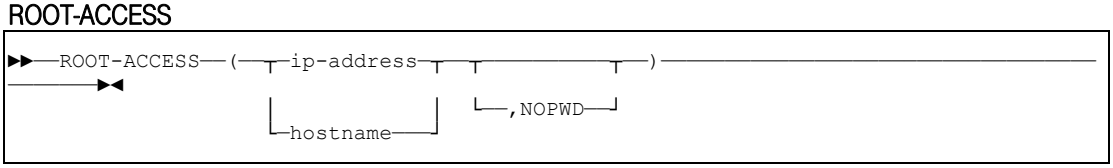
Restriction: The ROOT user always needs a password regardless of the TRUSTED-HOST definitions. Furthermore the requesting host must be defined as ROOT-ACCESS host.

ip-address ip address in dotted format.

hostname Hostname of a host.
While startup the hostname will be resolved therfore the hostname must be defined in /etc/hosts or in a DNS.

ACCAPIUID Flag indicating that the server accepts a userid specified via the NCI API as trusted. If this flag is omitted the server only accepts a userid as trusted if the userid has been added from the NCI internal client code.
Note: NCI internal client code automatically adds the effective userid of the process to the NCI data stream if no userid is given via the NCI API.

ROOT-ACCESS



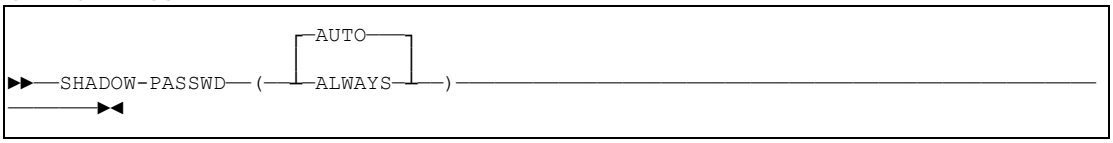
Parameter specifies hosts allowed to access the server with a root userid.

ip-address ip address in dotted format.

hostname Hostname of a host.
While startup the hostname will be resolved therfore the hostname must be defined in /etc/hosts or in a DNS.

NOPWD Flag indicating, that the root user can access TP's with security level PROPAGATE without specifing a password, if the NCI client is invoked with uid=0. The default expects a password from a client invoked with uid 0. This flag doesn't affect the behaviour of the user parameter set by the NCI API. It takes only effect on the uid of the invoker of the client.

SHADOW-PASSWD

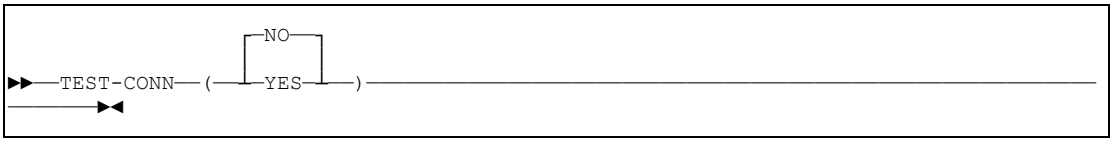


Parameter is currently only used on Linux and Sun SOLARIS systems and controls the use of a shadow password file for user verification.

AUTO Autodetect precence of a shadow password file. Server uses the shadow password file, if the password entry in the /etc/passwd file contains an 'x'.

ALWAYS /etc/passwd is not checked. Only shadow password file is used for password verification. If a shadow password file is not present the verification fails and the client connection will be rejected.

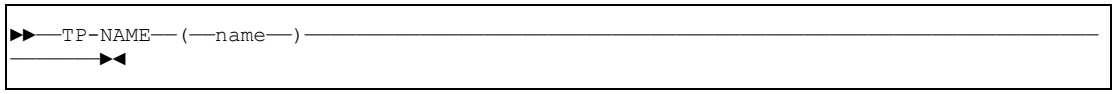
TEST-CONN



If this parameter is set to YES, the tcpip connection will be tested before a send() will be performed. The default for this parameter is NO. Set this parameter to YES could increase communication time (on UNIX typically 1-3 ms).
This feature had to be introduced due to the design of TCP/IP.

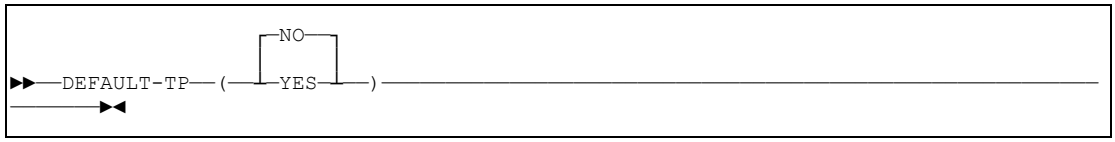
3.1.3.2 TP Specific Parameters

TP-NAME



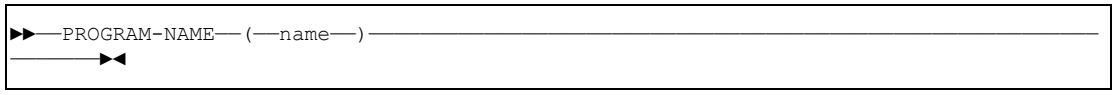
name Application-specific TP-Name (1 to 60 characters) that identifies a NCI Transaction Program and its properties.

DEFAULT-TP



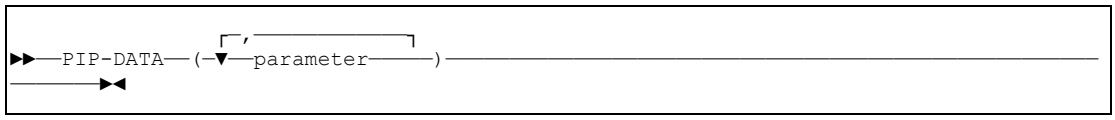
Only one TP can be defined as DEFAULT-TP. If a Client requests a service without specifying a TP-Name, the default TP-Definitions are used. If no TP-Name is defined as default, the client request will be denied by the NCI Communication Manager.

PROGRAM-NAME



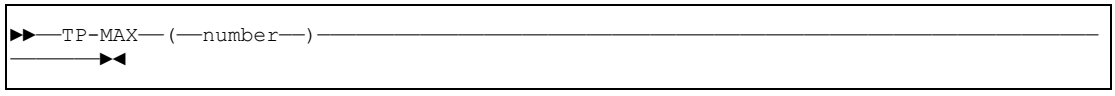
name Application Program Name that should be started by the NCI Communication Manager to process the client request. Referring by an absolute path is possible.

PIP-DATA



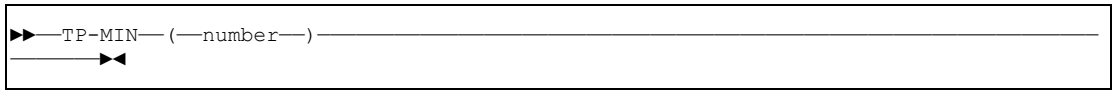
parameter Program Initial Parameters (1 to 100 characters) that the application program will receive, if it is started for the first time.
Each paramter must be separated by a comma. The PIP data will be passed to the application program in the same format as if the program had been invoked from the command line.

TP-MAX



number Number between 1 and 999 specifying the maximal number of TPs that should be started. If the maximum number of started TPs is reached, all subsequent requests are queued until a TP becomes available.

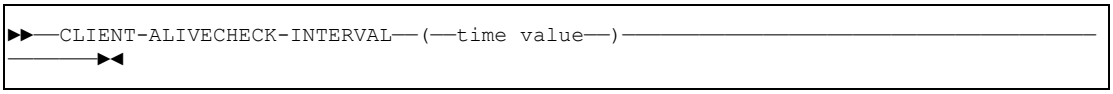
TP-MIN



number Number between 1 and 99 specifying the number of TPs that should be started while

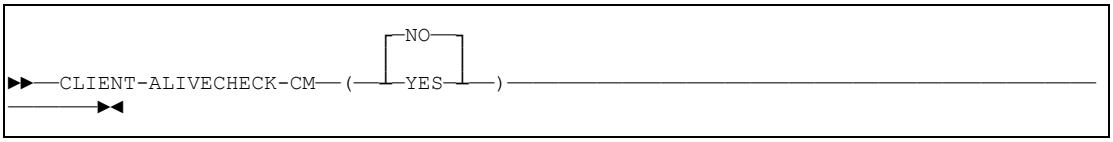
NCI Communication Manager startup.
Note: TPs started during startup are not affected by idle time expiration. That means they remain active until server shutdown.

CLIENT-ALIVECHECK-INTERVAL



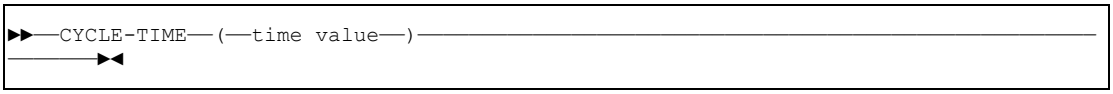
time value Defines the interval time (in minutes) between two "Client Alive Checks". The "Client Alive Check" is only done while a TP is working for a client. If the alive check determines that the client closed the connection the TP raises the signal SIGKILL and the TP will be terminated. If the parameter is omitted no "Client Alive Check" is done. Please consider, monitoring the client connections consumes additional resources and should only be activated, if your application needs this feature.

CLIENT-ALIVECHECK-CM



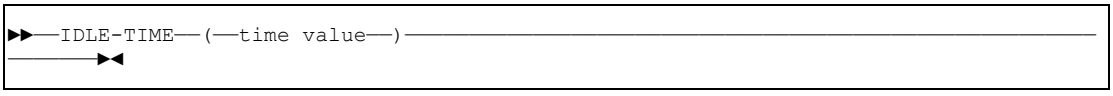
If this parameter is set to 'YES' the "Client Alive Check" for the TP is done from the NCI Communication Manager rather than the TP itself. This is maybe necessary if the check can't be performed by the TP. The TP uses the alarm timer and a SIGALRM handler to periodically do the check. However the SIGALRM signal may interrupts the application and the application may not recover and gets and error. In this case the check must be done by the Communication Manager. This option should be only used if required, because the check needs additional resources.

CYCLE-TIME



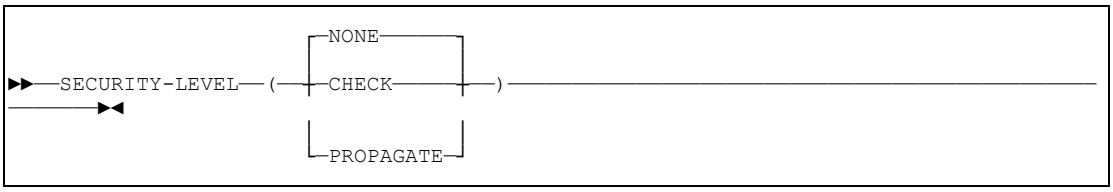
time value Time in minutes between 0 and 1440. The cycle time elapses periodically and may be used from the application to do periodic work.
Each time the cycle time elapses NCI returns control to the application ith program with rc=12.

IDLE-TIME



time value Time in minutes between 0 and 1440. The idle time is counted each time a TP has no more work to do. If a TP is idle for the specified time NCI returns control to the application with rc=8. The application program should terminate in this case.
Note: For UNIX this parameter is not valid for TPs defined with a SECURITY-LEVEL of PROPAGATE. TPs defined with a security level of propagte will be immediately stopped after they have processed the client request.

SECURITY-LEVEL

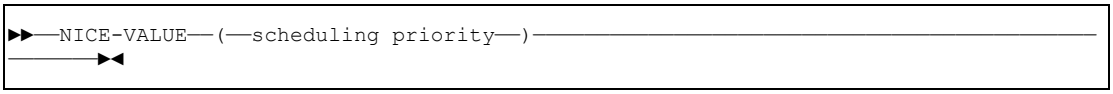


NONE NCI Communication Manager does not perform any security checks. The client's userid and group can be accessed by the server application after function code: RECEIVE, assuming the client supplies security information.

CHECK The client's allocation request must provide a valid userid and password on server side, otherwise the request will be denied.
Exception: The server also accepts requests without a password from those clients defined as already verified (see TRUSTED-HOST).
If protocol TCP/IP is used: attribute already verified can be restricted to those hosts defined with the TRUSTED-HOST parameter.

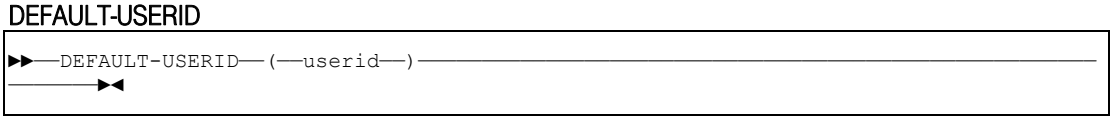
PROPAGATE Implies CHECK, with the extension that the client userid is propagated to the TP. That means the TP runs under the authority of the client user.

NICE-VALUE



scheduling priority this parameter allows to lower the schedule priority
0 = highest prio, same as communication manager itself
5 = default
19 = lowest prio, batch priority on most systems
Set the priority on a value less than 5, i.e. higher prio, the internal TP's or the communication manager could be not scheduled in an optimal way

DEFAULT-USERID



userid If you want your TPs to run under a userid different from the one used by the NCI Communication Manager process, this userid may be specified here.

Note: The NCI Communication Manager process must run with root authority if at least one TP has been defined with one of the parameters listed below.

- SECURITY-LEVEL(CHECK)
- SECURITY-LEVEL(PROPAGATE)
- DEFAULT-USERID(...)

3.1.3.3 Sample Configuration File

```
************************************************************
* NCI CONFIGURATION DATA FOR APPLICATION: TEST             *
************************************************************
*
* Terminate the Communication Manager after last TP stops.
  AUTO-TERMINATION(YES)
*
* Use IP-PORT 3450 to listen for client requests.
  IP-PORT(3450)
*
* Use the number 5555 to allocate a shared memory segment
  SHM-KEY(5555)
*
* Write Error Messages to stdout.
  MSG-LOG(1)
*
* Write Trace Messages to the file /u/hugo/nci.trace.log
  TRACE-FILE(/u/hugo/nci.trace.log)
*
* Set Trace level
* Trace all queued requests and starting and stopping of TPs
  TRACE(QUE,CSS)
*
*+++ TP definitions +++++++
*
    TP-NAME(APP1)                 .ANY CHAR 1-60
      DEFAULT-TP(NO)              .No default TP
      PROGRAM-NAME(app1xyz)       .program-name 1-8
      TP-MIN(3)                   .start 3 TPs at startup
      TP-MAX(10)                  .start max. 10 TPs
      IDLE-TIME(10)               .stop TP if more than 10 minutes
*                                 . idle
      CYCLE-TIME(2)               .wake up TP each 2 minutes
      SECURITY-LEVEL(CHECK)       .check clients userid and passw.
*
    TP-NAME(APP2)                 .ANY CHAR 1-60
      DEFAULT-TP(NO)              .No default TP
      PROGRAM-NAME(app2pgm)       .program-name 1-8
      PIP-DATA(p1,p2,'p3 p3 p3')  .program initial parameters
      TP-MAX(2)                   .start max. 2 TPs
      SECURITY-LEVEL(PROPAGATE)   .check clients userid and passw.
*                                 .run the TP under the clients
*                                 . userid
```

Figure 3-2: Sample NCI Configuration File for UNIX

3.2 NCI Communication Manager for Windows (Win32)

3.2.1 General Introduction

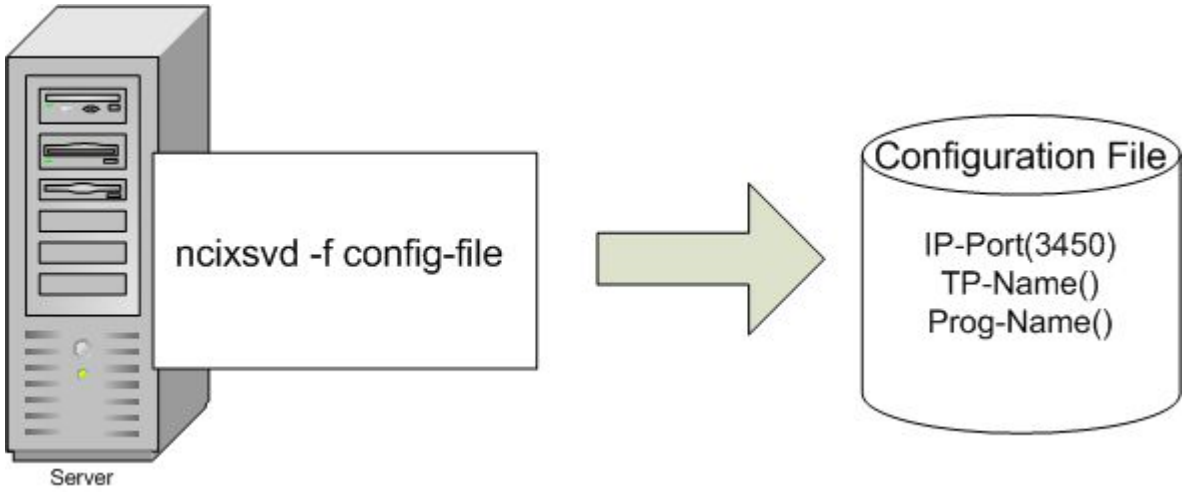


Figure 3-3: NCI Communication Manager for Windows (Win32)

The NCI Communcation Manager could be started either as a command line call or by a API call. All the necessary configuration information which is needed to run the NCI Communication Manager is located in a configuration file. These configuration data could the TCP/IP Port information, which Ethernet Interface it is listening to, the TP Program information, and a lot more. To start the NCI Communcation Manager it is absolutely necessary to provide this configuration file, independent of the way it is started.

3.2.2 Descrpition of the NCI Communication Manager interface

The NCI Communication Manager accepts two kind of interfaces:

1. Program-Call interface
   NCI Communication Manager may be called from an application program as a subroutine, contained in the nci shared library.

```
#include "nci.h"

 void main(void)
{
int rc;
  /* perform some application specific processing */
  doSomething();

  /* call nci communication manger */
  rc = ncixcm(.....);

  return 0;
}
```

2. Command Line interface
   Front end to call the NCI Communication Manager from the command line. This front end only

interprets the passed commandline parameters and then calls the nci Communication Manager library routine.

```
$ ncixsvd -f config-file -d
```

Parameters

NCIXCM

The library routine accepts as parameter only a pointer to a parameter string. This parameter string is a string of ncixcm parameter keywords each keyword separated by a semicolon. The following keywords are valid.

- CONFIG-FILE(name)
  Name of the nci Communication Manager configuration file. This file may contain additional NCI configuration parameters.
- all keywords which are valid for the NCI configuration file.

Example

```
#include "nci.h"

void main(void)
{
  int rc;
  char *params="CONFIG-FILE(c:\\etc\\nci.cfg);IP-PORT(3450);..."

  rc = ncixcm(params);

  exit
}
```

Note: If a keyword is specified in the configuration file and is additionally provided by the parameter string, the keyword from the parameter string takes precedence over the keyword in the configuration file.

Command Line interface

The command line interface accepts the following parameters.

-h	Display information about supported command line parameters.
-d	Daemonize the process. If this switch is used the process will run as a daemon.
-f configurationfile	NCI configuration file to be used by the server. If this parameter is omitted a file named <ncixsvd.cfg> in the current working directory is expected.
-s ipport	TCP/IP port number to be used by the server. This flag overwrites the IP-PORT definition within the configuration file.
-w working directory	Working directory to be used by the server daemon. If this parameter is omitted the default root directory '/' is used. This parameter will only be used if the paramter -d is also specified.

Example

```
$ ncixsvd -d -w c:\home\hugo -f hugonci.cfg
```

In the example listed above the Communication Manager process will be daemonized. The working directory is changed to *c:|home|hugo* and the name of the nci configuration file is *hugonci.cfg*.

3.2.2.1 System Service

NCI Communication Manager may be used as a Windows System Service. It is automatically installed by the Windows 2000/XP nci setup procedure. For installing System Services see Windows 2000/XP administration manuals. The name of the service program is nciservice.exe. It launches two tasks. First it registers itself at the Service Manager, and as a second step it interprets the passed commandline parameters and then calls the nci Communication Manager library routine. Except of the Service Manager registration and additional System Service handling it has the same functionality as the Command Line interface.

Note: This service could not run on Windows NT systems. The minimum requirement is Windows 2000/XP.

Parameters

The NCI System Service accept only one parameter.

-f configurationfile NCI configuration-file to be used by the server. If this parameter is omitted a file named <ncixsvd.cfg> in the current working directory is searched.

Search Order

In the case of using NCIService the directory where the logfiles are created as well as executables are searched if they are not located in the PATH directories can be determined by the following procedure.

1. if the environment variable NCI_SERVICEDIR is set, this directory is used
2. the path to the location of the service module is used, if it is fully qualified
3. if the environment variable TEMP set, this directory is used
4. if 1-3 failed the directory C:\ is used.

In this release of NCICM for Windows 2000/XP the log and trace files are configured in the service config file. Another log file called nciservice.log can be found there, it contains only log messages of the Windows Service Dispatcher.

Note: The TP's have to be in a directory which is listed in the PATH Variable or in the directory where the NCI/CM has been started. In the case of NCIService the current directory is determined by the above described procedure.

Security

If security mechanisms are of interest, it is necessary to give the owner of the NCI Communication Manager the Privilege: *Owner is part of the trusted computer base (SeTcbPrivilege).* This privilege is granted automatically, if the service is normally installed, e.g. it is running with System authority.

ServiceManager

To handle the Windows System Services a little bit more easier, a small Tool is added to the NCI Package: ServiceManager.

Type	Description	
Module:	the name of the executable with full pathname, the '>>' button opens a file dialog	
Arguments:	if the service supports arguments they can passed here, for example an apropraie config file	
Display:	the name under which the service should be listed in the Service Menu of the WinNT System Panel, if this is omitted, the Display name is the same as the name of the Service	
Service:	the name of the service, under this name the OS refers to it	
StartType:	*manually*	the service is managed manually
	automatic	the service is started at system startup, as well the ServiceManager tries to start teh service after installing
	disable	the service will be installed, but will remain disabled, it cannot be managed
Host:	name of remote machine, where the services can be managed, if nothing is supplied the local machine is used	
Install:	install the service. This have to be done before any other action	
Remove:	the service from the system. if necessary service will be stopped automatically.	
Start:	manuall service start. The service process is loaded into memory and is running.	
Stop:	manuall service stop. The service is removed from the active processes and from memory.	
Suspend:	the service will be suspended, it remains in the address space, but the service is inactive	
Continue:	continue a suspended service	
Query:	query the state of a service	
Exit:	exit ServiceManager	

Table 3-2: NCI Service Manager for Windows Options

Note: If any parameter has to be changed, first the service has to be removed, then the parameters can be changed. Only after a new installation of the service the changed parameters take effect.

To install the NCIService the following parameters have to be supplied:

Example

Entry	Example
Module:	C:\...\nciservice.exe
Arguments:	-fC:\...\nciservice.cfg
Display:	Test nci
Service:	nci

If the entry *Arguments* is omitted the default configuration <ncixsvd.cfg> in the current work directory is used. To determine the current work directory, see above.

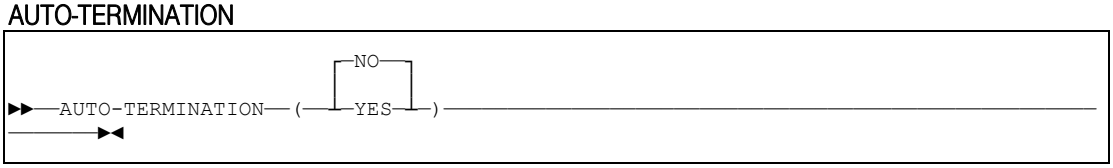
Now the service can be installed by pressing Install. If the ServiceType was manully the service can be started by pressing Start. If the ServiceType was automatic, the ServiceManager has tried to start the service already after successfull installation.

3.2.3 Configuration File

Each NCI server needs a separate configuration file. In contrast to the OS/390 platform were the nci server configuration is devided into two parts, a configuration module and a configuration file, on the UNIX and Windows NT platform all server configuration parameters are contained in the configuration file.
The configuration parameters can be divided into two categories. Global parameters used to control the characteristics of the whole server and TP specific parameters used to define a TP entry.

3.2.3.1 Global Parameters

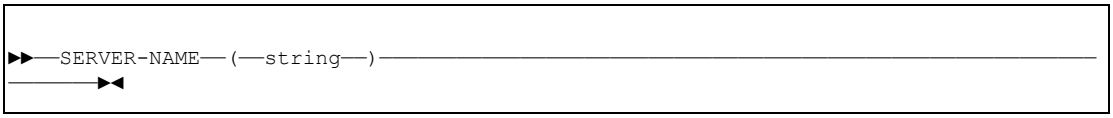
AUTO-TERMINATION



YES Terminate the server after the last TP ended

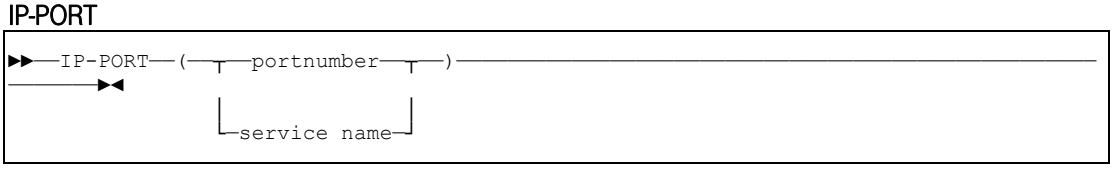
NO Server runs until TERM signal received.

SERVER-NAME



string Name (1-20 characters) of the application.
The string is used for NCI internal control block eye catchers and as syslog message prefix.

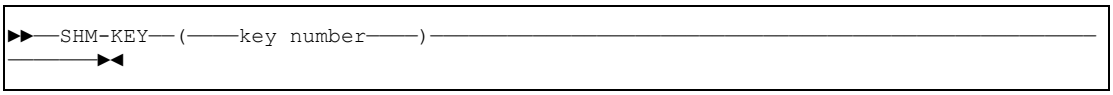
IP-PORT



port number IP port to be used by the NCI Communication Manager.

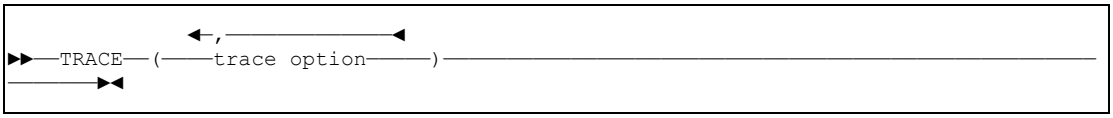
servicename Name of a service defined in the C:\WINNT\system32\drivers\etc\services file to be used by the NCI Communication Manager. While startup NCI Communication Manager uses the getservbyname call to obtain the port number corresponding to the defined service name.

SHM-KEY



key number Numeric digit specifying the key to be used to create a shared memory segment. Must be unique over the whole system !

TRACE



Controls the level of trace messages written to the NCI trace file. If this parameter is specified the TRACE-FILE parameter must also be specified.

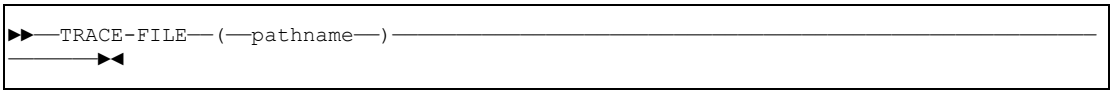
Note: Due to performance impacts tracing should be only activated for error analysis.

Valid trace options are:

NONE	**Disable Trace.**
ALL	Trace all.
FUNC	Function Trace.
DATA	Tracing of data flow
OPTIONS	NCI options being used.
CB	Tracing of NCI Control Blocks
CODESET	Tracing of Character Set handling
PROT	Specific transport protocol trace
QUE	Trace of request queuing
CMC	Trace of parallel server communication
CSS	Trace of parallel server handling
API	Trace of special API calls (e.g. RFC)
DYN	Trace dynamic loading
ACCT	Trace of accounting
SEC	Trace of security relevant settings

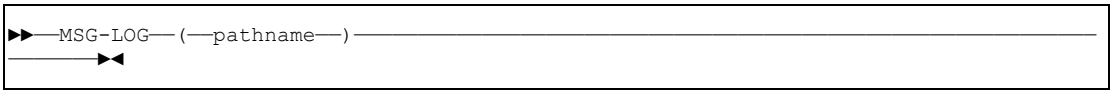
DATACNV Trace Data Conversion

TRACE-FILE



pathname Path and filename of NCI trace file.
To write trace messages to an already open file descriptor eg. stdout, the output stream may be specified as numeric digit.

MSG-LOG



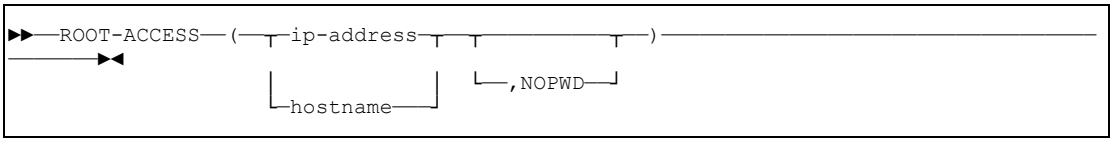
pathname Path and filename of NCI message log file.
To write trace messages to an already open file descriptor eg. stdout, the output stream may be specified as numeric digit.

Note: When configuring a Windows System Service, do not use *nciservice.log* as a log or tracefile, because it is reserved for the service main tasks log.

TRUSTED-HOST

Not supported in Win32.

ROOT-ACCESS



Parameter specifies hosts allowed to access the server with a Administrator userid.

ip-address ip address in dotted format.

hostname Hostname of a host.
While startup the hostname will be resolved therfore the hostname must be defined in C:\WINNT\system32\drivers\etc\hosts or in a DNS

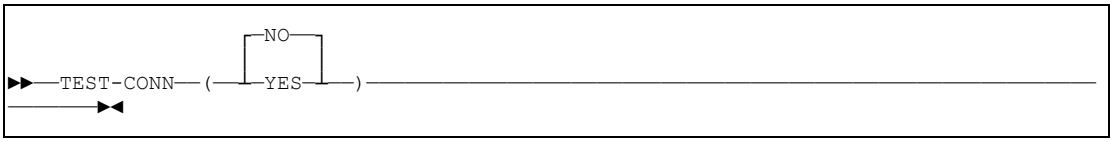
NOPWD Flag indicating, that the Administrator can access TP's with security level PROPAGATE without specifing a password, if the NCI client is invoked by an Administrator account. The default expects a password from a client invoked by the Administrator. This flag doesn't affect the behaviour of the user parameter set by the NCI API. It takes only effect on the uid of the invoker of the client.

Note: Under Windows 2000/XP this refers explicitely to the login: *Administrator*. No other account is checked regardless of its privileges.

SHADOW-PASSWD

Not supported in Win32.

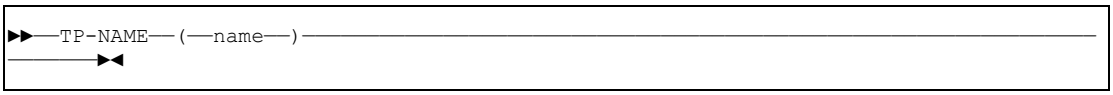
TEST-CONN



If this parameter is set to YES, the tcpip connection will be tested before a send() will be performed. The default for this parameter is NO. Set this parameter to YES could increase communication time (on UNIX typically 1-3 ms).
This feature had to be introduced due to the design of TCP/IP.

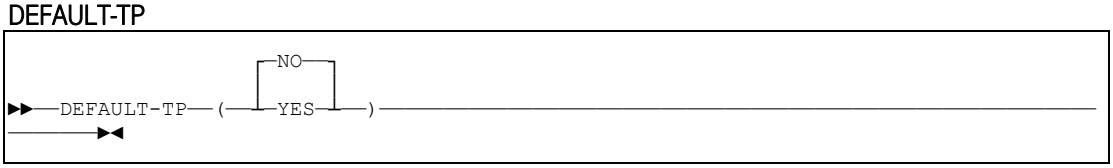
3.2.3.2 TP Specific Parameters

TP-NAME



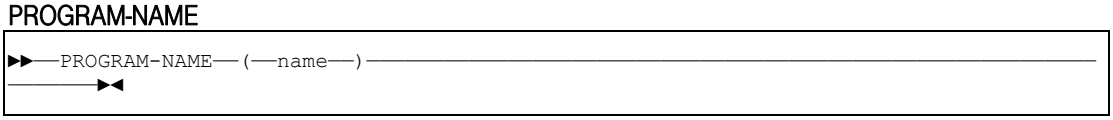
name Application-specific TP-Name (1 to 60 characters) that identifies a NCI Transaction Program and its properties.

DEFAULT-TP



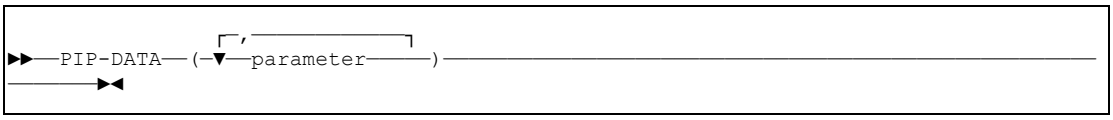
Only one TP can be defined as DEFAULT-TP. If a Client requests a service without specifying a TP-Name, the default TP-Definitions are used. If no TP-Name is defined as default, the client request will be denied by the NCI Communication Manager.

PROGRAM-NAME



name Application Program Name that should be started by the NCI Communication Manager to process the client request. Referring by an absolute path is possible. The suffix *.exe* **must** be omitted. It will be appended automatically by the communication manager.

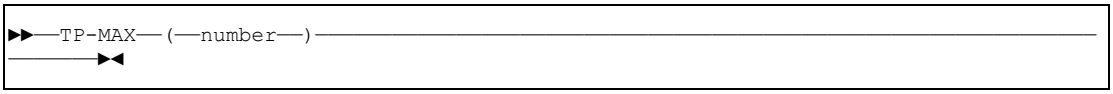
PIP-DATA



parameter Program Initial Parameters (1 to 100 characters) that the application program will receive, if it is started for the first time.

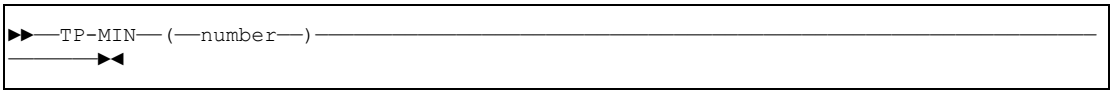
Each paramter must be separated by a comma. The PIP data will be passed to the application program in the same format as if the program had been invoked from the command line.

TP-MAX



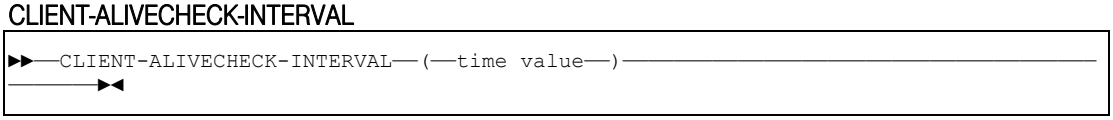
number Number between 1 and 999 specifying the maximal number of TPs that should be started. If the maximum number of started TPs is reached, all subsequent requests are queued until a TP becomes available.

TP-MIN



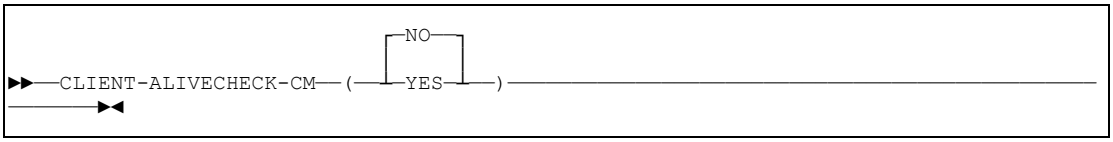
number Number between 1 and 99 specifying the number of TPs that should be started while NCI Communication Manager startup.
Note: TPs started during startup are not affected by idle time expiration. That means they remain active until server shutdown.

CLIENT-ALIVECHECK-INTERVAL



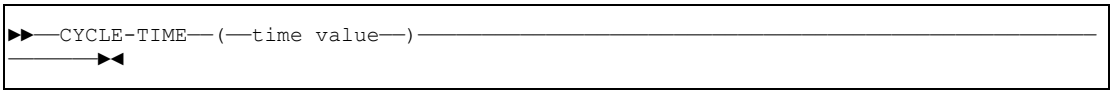
time value Defines the interval time (in minutes) between two "Client Alive Checks". The "Client Alive Check" is only done while a TP is working for a client. If the alive check determines that the client closed the connection the TP raises the signal SIGKILL and the TP will be terminated. If the parameter is omitted no "Client Alive Check" is done. Please consider, monitoring the client connections consumes additional resources and should only be activated, if your application needs this feature.

CLIENT-ALIVECHECK-CM



If this parameter is set to 'YES' the "Client Alive Check" for the TP is done from the NCI Communication Manager rather than the TP itself. This is maybe necessary if the check can't be performed by the TP. The TP uses the alarm timer and a SIGALRM handler to periodically do the check. However the SIGALRM signal may interrupts the application and the application may not recover and gets and error. In this case the check must be done by the Communication Manager. This option should be only used if required, because the check needs additional resources.

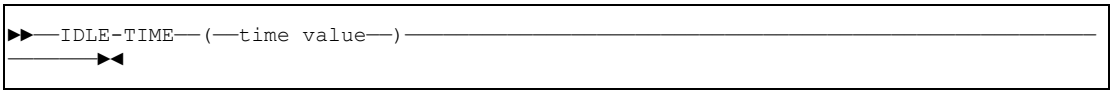
CYCLE-TIME



time value Time in minutes between 0 and 1440. The cycle time elapses periodically and may be

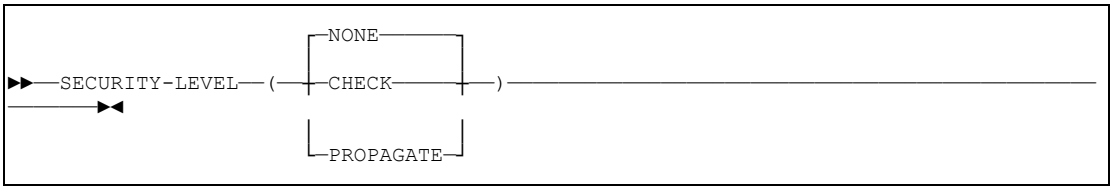
used from the application to do periodic work.
Each time the cycle time elapses NCI returns control to the application ith program with rc=12.

IDLE-TIME



time value Time in minutes between 0 and 1440. The idle time is counted each time a TP has no more work to do. If a TP is idle for the specified time NCI returns control to the application with rc=8. The application program should terminate in this case.
Note: For Windows this parameter is not valid for TPs defined with a SECURITY-LEVEL of PROPAGATE. TPs defined with a security level of propagte will be immediately stopped after they have processed the client request.

SECURITY-LEVEL



NONE NCI Communication Manager does not perform any security checks. The client's userid and group can be accessed by the server application after function code: RECEIVE, assuming the client supplies security information.

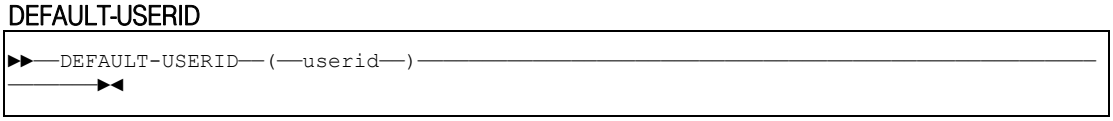
CHECK The client's allocation request must provide a valid userid and password on server side, otherwise the request will be denied.
Exception: The server also accepts requests without a password from those clients defined as already verified (see TRUSTED-HOST).
If protocol TCP/IP is used: attribute already verified can be restricted to those hosts defined with the TRUSTED-HOST parameter.

PROPAGATE Implies CHECK, with the extension that the client userid is propagated to the TP. That means the TP runs under the authority of the client user.

NICE-VALUE

Not supported in Win32.

DEFAULT-USERID



userid If you want your TPs to run under a userid different from the one used by the NCI Communication Manager process, this userid may be specified here.

Note: The NCI Communication Manager process must run with root authority if at least one TP has been defined with one of the parameters listed below.

- SECURITY-LEVEL(CHECK)

- SECURITY-LEVEL(PROPAGATE)
- DEFAULT-USERID(...)

3.2.3.3 Sample Configuration File

```
*********************************************************
* NCI CONFIGURATION DATA FOR APPLICATION: TEST           *
*********************************************************
*
* Terminate the Communication Manager after last TP stops.
  AUTO-TERMINATION(YES)
*
* Use IP-PORT 3450 to listen for client requests.
  IP-PORT(3450)
*
* Use the number 5555 to allocate a shared memory segment
  SHM-KEY(5555)
*
* Write Error Messages to stdout.
  MSG-LOG(1)
*
* Write Trace Messages to the file c:\home\hugo\ncitrace.log
  TRACE-FILE(c:\home\hugo\ncitrace.log)
*
* Set Trace level
* Trace all queued requests and starting and stopping of TPs
  TRACE(QUE,CSS)
*
*+++ TP definitions +++++++
*
    TP-NAME(APP1)                 .ANY CHAR 1-60
      DEFAULT-TP(NO)              .No default TP
      PROGRAM-NAME(app1xyz)       .program-name 1-8
      TP-MIN(3)                   .start 3 TPs at startup
      TP-MAX(10)                  .start max. 10 TPs
      IDLE-TIME(10)               .stop TP if more than 10 minutes
*                                 . idle
      CYCLE-TIME(2)               .wake up TP each 2 minutes
      SECURITY-LEVEL(CHECK)       .check clients userid and passw.
*
    TP-NAME(APP2)                 .ANY CHAR 1-60
      DEFAULT-TP(NO)              .No default TP
      PROGRAM-NAME(app2pgm)       .program-name 1-8
      PIP-DATA(p1,p2,'p3 p3 p3')  .program initial parameters
      TP-MAX(2)                   .start max. 2 TPs
      SECURITY-LEVEL(PROPAGATE)   .check clients userid and passw.
*                                 .run the TP under the clients
*                                 . userid
```

Figure 3-4: Sample NCI Configuration File for Win32

4 Sideinformation

NCI sideinformation can be used to provide application specific control options, object names and addressing information outside the application. If an application wants to use sideinformation, the following API calls must be used:

- NCISetAddrType(SIDE)
- NCISetPrimAddrInfo(name of the sideinformation file/module)
- NCISetSecAddrInfo(symbolic entry name)

The NCI sideinformation file contains a list of keywords (parameters) as descriped later on.
Each keyword must begin and end on a separate line. All data between the open brace and closing brace is treated as the keyword value. Leading or trailing blanks are not stripped !

Note: Keyword identifiers are not case sensitive. However the keyword values are all case sensitive. To prevent errors, code the keyword values in exactly the same syntax they are described in the reference chapters.

Lines starting with a '*' are treated as comment lines.

Each NCI sideinformation file must contain at least one named symbolic section (keyword SymbolicName). NCI doesn't support a default section. A sideinformation file may contain multiple different sections, offering the flexibility to define different application definitions within the same sideinformation file.

All keywords defined in the sideinformation file before the first symbolic section name are treated as global keywords. Global keywords are propagated to all subsequent symbolic sections.

Note: If a global keyword is also defined within a symbolic section, the global definitions will be overridden by the definitions in the symbolic section.

```
AddrType(MQ)              // adddressing type: message queue
   PrimAddrInfo(QMGR1)       // default queue manager name
   SecAddrInfo(DFLTQ)        // default queue name
   SyncpointOpt(N)           // no syncpoint processing

SymbolicName(LQ1IN)
   SecAddrInfo(Q1)           // queue name
   ServiceId(MSG1)           // filter out messages with ServiceId MSG1
   KeepConnection(Y)         // open queue only once and keep it opened
   Timeout(600)              // message waittime: 10 minutes
   DataConv(Y)               // convert data (ASCII/EBCDIC)
   SideApplData(PRM1,PRM2)   // data that can be accessed by the appl.
   ErrorMsgOpt(STDERR)       // write error messages to stdout
   TraceOpt(ALL)             // activate full function trace


SymbolicName(LQ1OUT)
   SecAddrInfo(Q1)           // queue name

SymbolicName(LQ2)
   SecAddrInfo(Q2)           // queue name
```

Figure 4-1: Example NCI sideinformation file

On OS/390 the sideinformation file must be converted to a sideinformation load module. An ISPF-Dialog will assist you to generate a NCI sideinformation module.

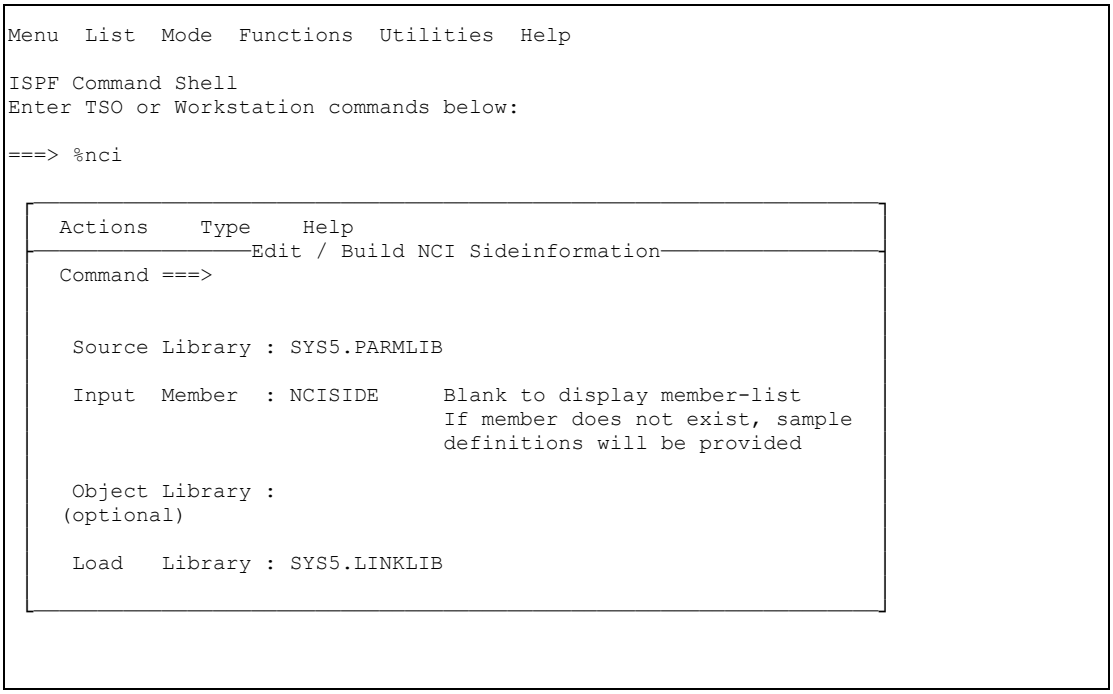


Figure 4-2: z/OS ISPF Dialog

4.1 Parameter description

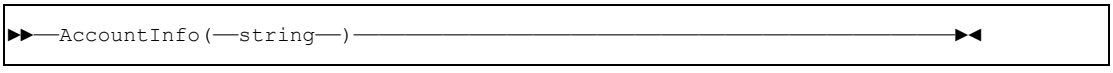


Figure 4-3: Accounting Information

Note: This keyword 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*.

Special behaviour for operating system OS/390:
If no accounting information will be provided by the application or via NCI sideinformation, the default accounting information of the current process (OS/390 address space) will be used.

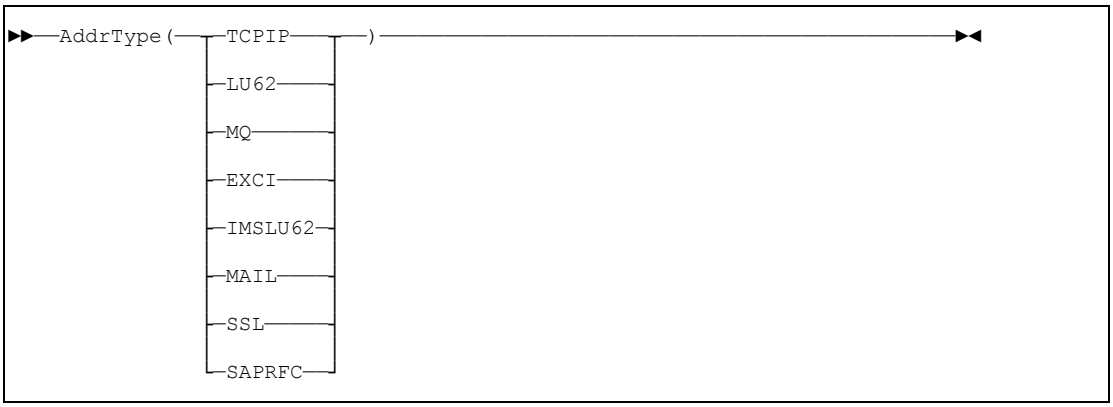


Figure 4-4: Addressing Type

The parameter *AddrType* defines the transport protocol to be used. The following addressing types are supported:

Type	Description
TCPIP	TCP/IP socket interface will be used as transport protocol. The parameters PrimAddrInfo and SecAddrInfo will be used to address the TCP/IP target system and application. PrimAddrInfo must provide the TCP/IP hostname or the IP-address in dotted format (e.g. 53.113.127.10). SecAddrInfo must provide the TCP/IP servicename (defined in /etc/services) or portnumber (e.g.3450).
LU62	SNA LU 6.2 interface will be used as transport protocol. The parameter SecAddrInfo must provide the SNA application-id (SNA LU-name).
MQ	MQSeries will be used as transport protocol. The parameters PrimAddrInfo and SecAddrInfo will be used to address the MQSeries queue manager and queue name. PrimAddrInfo must provide the name of the MQSeries queue manager, otherwise the default queue manager will be used. SecAddrInfo must provide the MQSeries queue name.
EXCI	The CICS EXCI interface will be used to connect to a CICS system and call the desired CICS program. The parameter PrimAddrInfo is optional in this case and may be used to assign a transaction name to the started CICS program. SecAddrInfo specifies the LU-Name of the CICS system and ServiceId the name of the CICS program. **Note**: AddrType EXCI is only supported on OS/390.
IMSLU62	SNA LU 6.2 interface will be used to invoke a IMS transaction program. The parameter SecAddrInfo must provide the LU-Name of the IMS region and ServiceId specifies the IMS transaction code. **Note**: AddrType IMSLU62 is only supported on OS/390.
MAIL	The SMTP protocol will be used to send data as the mail body to a SMTP server. The parameter *PrimAddrInfo* must provide the IP address of the SMTP server. The parameter *ServiceId* must provide the mail receiver. For other necessary email parameters see the section on nciSetParam.
SSL	
SAPRFC	The SAP RFC (Remote function call) will be used to invoke a SAP ABAP function module. PrimAddrInfo must provide the SAP system name. This is used only for documentation and log entries. SecAddrInfo must provide the complete addressing information or "DEST=xxx"

where xxx is an entry in the optional saprfc.ini file. ServiceId specifies the SAP ABAP function module name.
Note: AddrType SAPRFC is only supported on Windows NT, SUN Solaris, HPUX, AIX, Linux, z/OS.

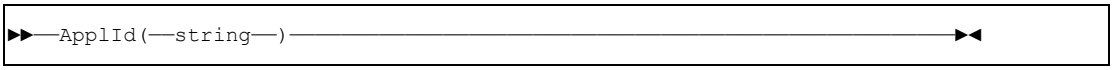


Figure 4-5: Application Identifier

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" in the “NCI Application Programming Reference”.

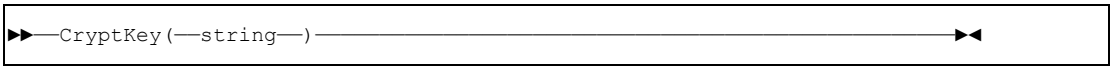


Figure 4-6: Cryptkey

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

Set the Encryption Key for the specified connection. It takes only effect if DataEncrypt is turned on. To support a format which is valid on all plattforms, the key has to be notated as a Hexadecimal number string where the 0x has to be ommitted, e.g. CryptKey(“0F564322378AB2”). The length of the key depends on the choosen encryption algorithm, see parameter “*DataEncryption*” for details.

Encryption algorithm	Necessary key length	Key string length
DES	8	16
TDES	24	48
QDES	40	80

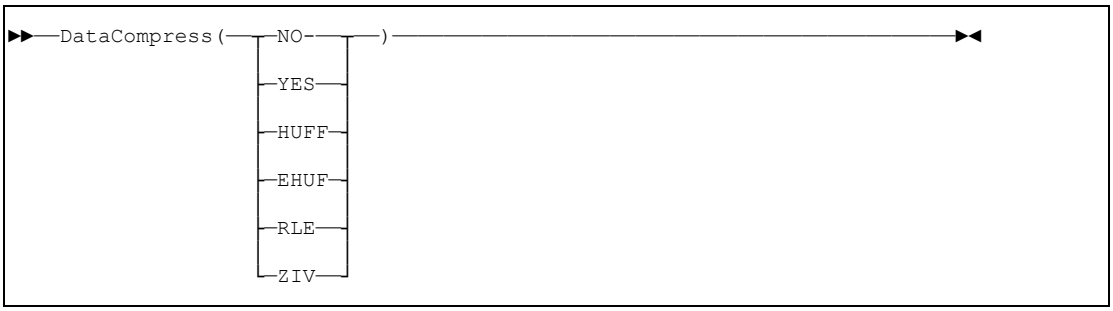


Figure 4-7: Data Compression

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

Option	Description
NO	No compression is requested for the specified connection.
YES	For compatibility reasons: Same as HUFF
HUFF	Compression is requested with Huffman algorithm.
EHUF	Compression is requested with Huffman algorithm with extra RLE encoding.
RLE	Compression is requested with RLE algorithm.
ZIV	Compression is requested with Lempel-Ziv algorithm.

Note: This feature is CPU consumptive.

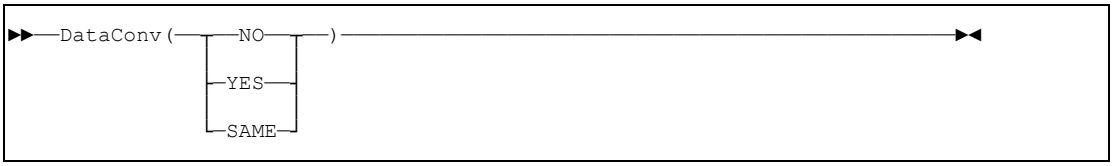


Figure 4-8: Data Conversion

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 has to be present in character format).

An application may request message data conversion via *DataConv(YES)*. If the partner application receives message data in network format, NCI will automatically convert the message data to host format before the server application gets control (after function: NCIGet).
Example: If a client application (running under Unix or Windows) sends message data in ASCII and option *DataConv(YES)* is in use, the data will be converted to EBCDIC, before the server application (running under OS/390) gets control (after function: NCIGet).
No data conversion will take place if the client sends the data with the option *DataConv(NO)*.

Behaviour for protocol TCP/IP:
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. Because XDR only supports 7 bit ASCII character sets, specific national characters with character code above 0x7f (127) shows a unpredictable behaviour.

Note: To obtain a conversion for the complete character set the new feature CodePage conversion is supported since NCI310. For details see parameter "*LocalCP*".

Behaviour for protocol MQSeries:
Data conversion will be done by MQSeries conversion routines.

Behaviour for protocol LU 6.2, EXCI and IMSLU62:
Data conversion is not supported

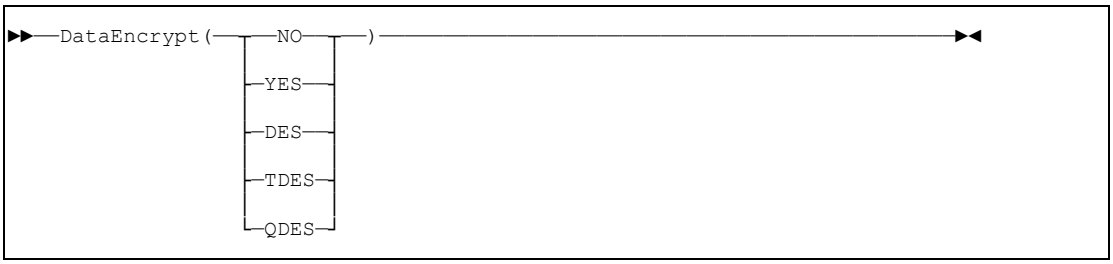


Figure 4-9: Data Encryption

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

Controls if message data should be encrypted before transmission.

Type	Description
NO	No encryption is requested for the specified connection.
YES	For compatibility reasons: Same as DES.
DES	Encryption is requested with DES algorithm. The encryption key needs 8 Byte, that is the hexadecimal string has to specifiy 16 hexadecimal numbers.
TDES	Encryption is requested with TRIPLEDES algorithm. The encryption key needs 24 Byte, that is the hexadecimal string has to specifiy 48 hexadecimal numbers.
QDES	Encryption is requested with MULTIPLE(5)-DES algorithm. The encryption key needs 40 Byte, that is the hexadecimal string has to specifiy 80 hexadecimal numbers.

Note: This feature is CPU consumptive.

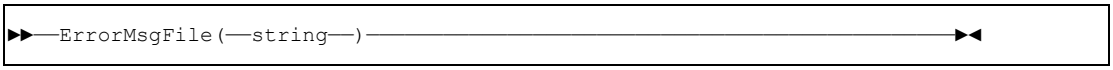


Figure 4-10: Error Message File

Note: This keyword is not supported on OS/390.

The default value for *ErrorMsgFile* is **ncierror.log**.
Defines the name of the file, where NCI error messages will be written if *ErrorMsgOpt(**FILE**)* is in effect.

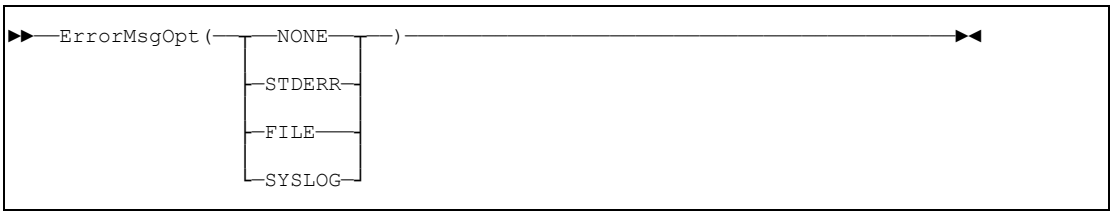


Figure 4-11: Error Message Processing Options

Controls the way error messages will be handled by NCI. The application can access NCI error messages (independent from *ErrorMsgOpt*) via function *NCIGetErrorMsgText*.

Value	Description
NONE	Error messages will be suppressed.
STDERR	Error messages will be written depending on the environment. *Env.* Description *OS* Error messages will be written to STDERR (on OS/390: Syslog). *CICS* Error messages will be written to CICS logfile. *IDMS* Error messages will be written to IDMS logfile.
FILE	Error messages will be written to an application-specific error message file specified by the NCISetErrorMsgFile function call or via NCI sideinformation. Note This option is not supported for OS/390
SYSLOG	Error messages will be written to the system log. Note This option is not supported for Windows

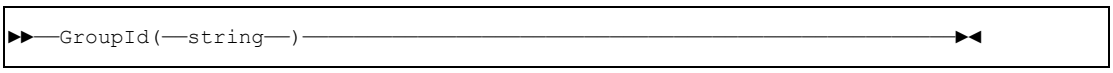


Figure 4-12: Group Identifier

Note: This keyword is not supported with protocols MQSeries, EXCI and IMSLU62.

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 and LU 6.2: 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.

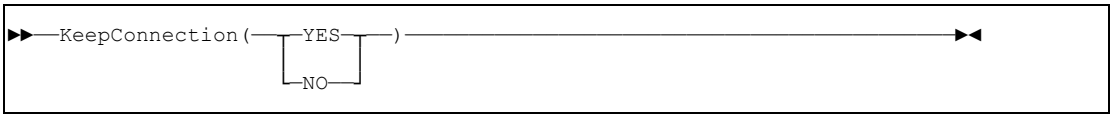


Figure 4-13: Keep Connection

Default value for protocols TCP/IP, LU62 and EXCI is **NO**.
Default value for protocol MQSeries and SAPRFC is **YES**.

The effect of the **SetKeepConnection** function call depends on the used protocol.

Behaviour for protocol TCP/IP and LU 6.2:
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.

Behaviour for protocol EXCI:
Controls whether a connection to the CICS system is kept open. Use this feature only if you want to make multiple putGet requests in succession. An open connection prevents the CICS system from terminating in an orderly manner.

Behaviour for protocol IMSLU62:
Not supported.

Behaviour for protocol SAPRFC:
Controls whether a connection to the SAP system is kept open. Use this feature only if you want to make multiple putGet requests in succession.

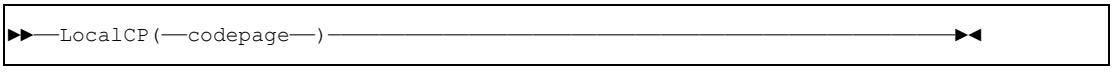


Figure 4-14: Local Code Page

Default value is unset.
The effect of the **LocalCP** function call depends on the used protocol.

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 has to be present in character format).

Behaviour for protocol TCP/IP:
Specifies the used Codepage on the local side. The effective Codepage is determined in the subsequently described way:

- Environment Variable (NCI_LOCALCP)
- API (nciSetLocalCodePage)
- Sideinfo configuration (LOCALCP)

Where Environment settings overwrite API settings, and API settings overwrite Sideinfo settings. Due to portability there are defined some platform independent names for a set of codepages, which have different names on some operating systems. This independent names start with a '**$**' char. If the codpage name parameter doesn't start with a '**$**' it is interpreted as the codepage name as it is provided by the operating system. There exist two special names: '**$STANDARD**' for compatibility with the old NCI conversion using XDR as described in nciSetDataConv and '**$AUTO**' where NCI tries to detect the systems default codepage.
In case of sending data the codepage will be used to interpret the sending data in this codepage and will be converted from this codepage into a internal format. In case of receiving data, the received data are converted from internal format to this codepage.

Note: It is necessary that the choosen codepage is installed correctly on the system.

Note: To use this codepage conversion, it is necessary to switch DataConversion to "**ON**" (see nciSetDataConv).

Note: To obtain a conversion for the complete character set the new feature CodePage conversion is supported since NCI310.

Value	Description
$STANDARD	For comptibility: uses old XDR conversion
$AUTO	Detect system codepage
$ISO88591	Set codepage to Latin-1 (ISO8859-1)
$ROMAN8	Set codepage roman8
$CP1252	Windows codepage, equivalent to Latin-1
$IBM1047	US EBCDIC page for z/OS USS
$IBM273	German EBCDIC for z/OS
<User defined>	The operating system name of any codepage.

Behaviour for protocol TCP/IP:
Controls the conversion of data.

Behaviour for protocol LU 6.2:
Not supported.

Behaviour for protocol MQSeries:
Not supported.

Behaviour for protocol EXCI:
Not supported.

Behaviour for protocol IMSLU62:
Not supported.

Behaviour for protocol SAPRFC:
Not supported.

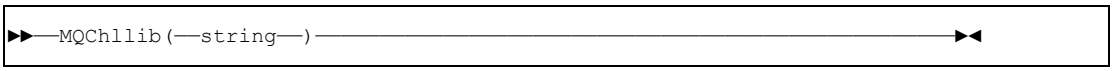


Figure 4-15: MQChillib Definition

Note: This keyword is only supported with protocol MQSeries.

Specifies the path to the directory containing the client channel definition table when using the MQSeries client. See the book **MQSeries Clients** for details.
If the **MQCHLLIB** environment variable has been specified, either through this keyword or outside the NCI application program it is assumed that the program will be executed in an MQSeries client environment.

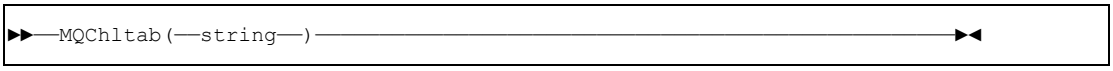


Figure 4-16: MQChaltab Definition

Note: This keyword is only supported with protocol MQSeries.

Specifies the name of the client channel definition table when using the MQSeries client. See the book **MQSeries Clients** by IBM for details.

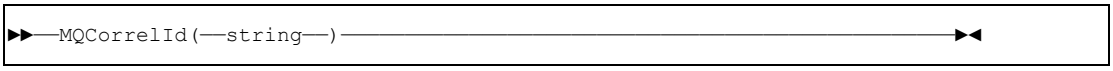


Figure 4-17: MQ Correlation Identifier

Note: This keyword is only supported with protocol MQSeries.

Specifies the correlation identifier for messages.
The effect of *setMQCorrelId* depends on the subsequent function calls.

NCIPut
If the put is a reply message to a previous request the correlation identifier is ignored. Otherwise a new message is created with this correlation identifier.

NCIPutGet
The correlation identifier is assigned to the request message, but is not used to retrieve the reply message.

NCIGet
Only messages with this specific correlation identifier are retrieved.

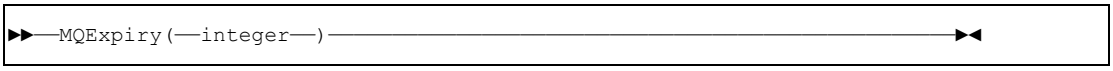


Figure 4-18: MQ Message Expiration Period

Note: This keyword is only supported with protocol MQSeries.

Specifies a period of time expressed in tenths of a second. The message becomes eligible to be discarded if it has not been removed from the destination queue before this period of time elapses.
A reply message is created with the same expiration period as the request message to which it belongs.

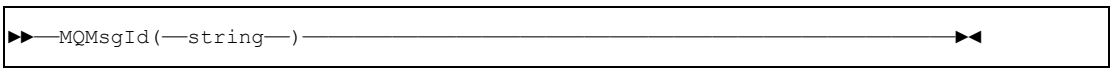


Figure 4-19: MQ Message Identifier

Note: This keyword is only supported with protocol MQSeries.

Specifies the message identifier for messages.
The effect of *setMQMsgId* depends on the subsequent function calls.

NCIPut
If the put is a reply message to a previous request the message identifier is ignored. Otherwise a new message is created with the message identifier.

NCIPutGet
The message identifier is assigned to the request message, but is not used to retrieve the reply message.

NCIGet
Only messages with this specific message identifier are retrieved.

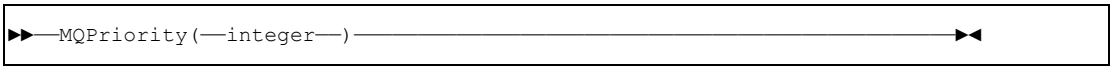


Figure 4-20: MQ Message Priority

Note: This keyword is only supported with protocol MQSeries.

Specifies the priority for new messages.
A reply message is created with the same priority as the request message to which it belongs.

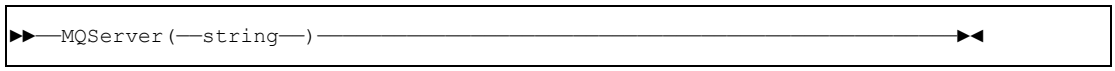


Figure 4-21: MQServer Definition

Note: This keyword is only supported with protocol MQSeries.

Specifies a minimal MQSeries client connection channel definition when using the MQSeries client, i.e. the location of the MQSeries server and the communication protocol to be used for connections.
The format of the MQServer string depends on the protocol (TCP/IP, LU62, Netbios, ...) used to connect the client to the server. For protocol TCP/IP the format is as follows:

ChannelName/TCP/ *HostName*/ *PortNumber*
For details see the book **MQSeries Clients** by IBM.
If the **MQSERVER** environment variable has been specified, either through this keyword or outside the NCI application program it is assumed that the program will be executed in an MQSeries client environment.

Note, that for NCI client security to be implemented, a client connection channel table, specified with the calls **NCISetMQChllib** and **NCISetMQServer**, must be used.

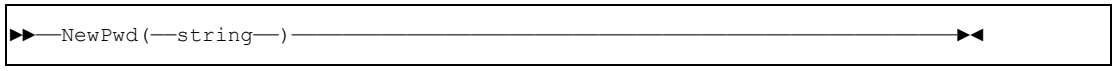


Figure 4-22: New Password

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 function *NCISetUserId* or NCI sideinformation is a valid RACF UserId.
- Password set by function *NCISetPwd* or NCI sideinformation is valid.
- NewPassword set by function *NCISetNewPwd* matches the RACF password rules.

Behaviour for protocol MQSeries, LU62, EXCI, IMSLU62 and SAPRFC:
Not supported.

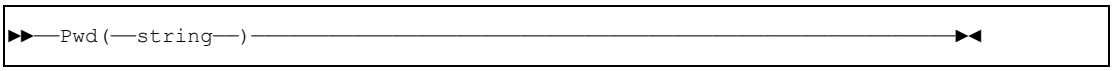


Figure 4-23: Password

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 and LU 6.2: 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.

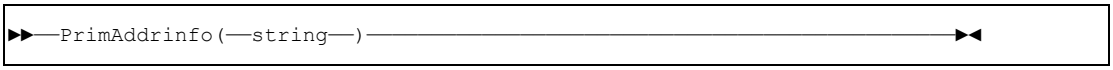


Figure 4-24: Primary Addressing Information

Depending on the parameter *AddrType* the parameter *PrimAddrInfo* has the following meaning:

Type	Description
TCPIP	For AddrType(TCPIP), the default value for PrimAddrInfo is 127.0.0.1 PrimAddrInfo must provide the TCP/IP hostname or the IP-address in dotted format (e.g. 53.113.127.10).
LU62	The parameter PrimAddrInfo is ignored.
MQ	For AddrType(MQ) PrimAddrInfo specifies the name of the MQSeries queue manager to which the application should connect. If PrimAddrInfo is omitted or specified as blank the default queue manager will be used.
EXCI	Id of the CICS mirror transaction under which the server program is to run. Only the first 4 characters will be used. If PrimAddrInfo is omitted "EXCI" is used as default.
IMSLU62	The parameter PrimAddrInfo is ignored.
SAPRFC	The parameter PrimAddrInfo specifies the name of the SAP target system (e.g. E1Z). It is used only for documentation and logging information.

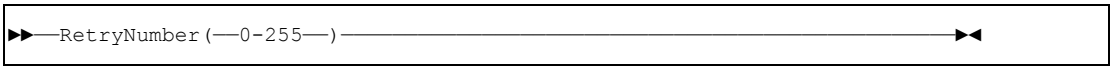


Figure 4-25: Retry Number

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 via function *nciSetRetryTime* or NCI sideinformation.

Connection errors for protocol TCP/IP and LU 6.2 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)

The function *nciSetRetryTime* is not supported with protocol EXCI.

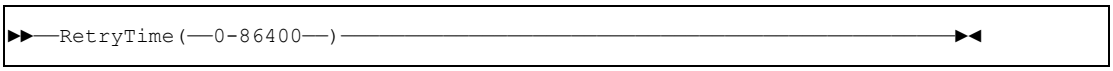


Figure 4-26: Retry Time

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.

function call *SetRetryNumber* or via NCI sideinformation.

Connection errors for protocol TCP/IP and LU 6.2 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)

The function call *SetRetryNumber* is not supported with protocol EXCI.

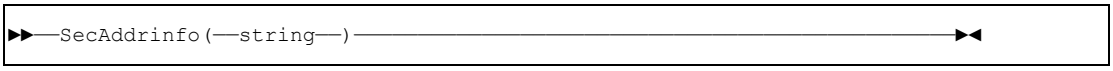


Figure 4-27. Secondary Addressing Information

Depending on parameter *AddrType* the parameter *SecAddrInfo* has the following meaning:

Type	Description
TCPIP	For AddrType(TCPIP), the default value for SecAddrInfo is 3450. SecAddrInfo must provide the TCP/IP servicename (defined in /etc/services) or portnumber (e.g.9432).
LU62	For AddrType(LU62), the default value for SecAddrInfo is APXXNCI. SecAddrInfo must provide the SNA application-id (SNA LU-name).
MQ	For AddrType(MQ) SecAddrInfo specifies the MQSeries queue name to be used on subsequent MQGet, MQPut or MQPutGet requests. The default value for SecAddrInfo is SYSTEM.DEFAULT.LOCAL.QUEUE.
EXCI	Applid of the CICS system to which the connect is to be done. Only the first 8 characters will be used. The default value for SecAddrInfo is APXXCICS.
IMSLU62	Applid of the IMS system to which the connect is to be done. Only the first 8 characters will be used. The default value for SecAddrInfo is APXXIMS.
MAIL	For AddrType MAIL, the default value for SecAddrInfo is 25. SecAddrInfo must provide the TCP/IP servicename (defined in/etc/services, typically smtp) or portnumber of the SMTP (e.g. sendmail demon) service on the SMTP Server.
SAPRFC	Complete addressing information for an SAP system which uses the format of the RfcOpenEx function from the RFC SDK of SAP. It is in the form of ID=value ID=value ... ID=value. For details see the RFC SDK, the most common are:

ASHOST	Hostname/IP Addr of SAP application server (mandatory)
SYSNR	R/3 Systemnumber (mandatory)
CLIENT	SAP logon client (mandatory)
DEST	Destination in saprfc.ini (using this key the file must exist)
TYPE	2/3 (Default is 3), describes the type of SAP system
LANG	SAP logon language, (i.e. DE for germany)
LCHECK	Logon check at open time
TRACE	0/1 (Default is 0), RFC Trace, 0=NOTRACE, 1=TRACE

There is no default value for SecAddrInfo for SAPRFC. Don't use USER= or PASSWD=, please use the corresponding NCI set calls.

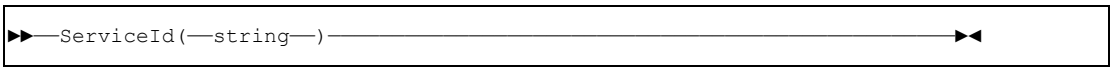


Figure 4-28: Service Identifier

The effect of **SetServiceId** depends on the used protocol.

Behaviour for protocol TCP/IP and LU 6.2:
The parameter *ServiceId* will only be used by the function call **NCIPutGet** to identify the transaction program that should be started by the NCI communication manager to handle this request.
Note: Only the first 60 characters are used as TP-Name.

Behaviour for protocol MQSeries:
For the function calls **NCIPut** and **NCIPutGet** the parameter *ServiceId* can be used to assign an identifier to a message being sent.
For the function call **NCIGet** the parameter *ServiceId* can be used as a search argument. Only messages with this specific ServiceId (set while **NCIPut** or **NCIPutGet**) will be retrieved from the message queue. *ServiceId* can be specified generic, i.e. with an asterisk as the last character. In this case all messages with a ServiceId identical to the one specified for the **NCIGet** call (without the asterisk) will be retrieved.
Note: Don't confuse this function with the message identifier set by **NCISetMQMsgId**. This function uses another field in the MQ message descriptor.
Only the first 28 bytes characters are used.

Behaviour for protocol EXCI:
Name of the CICS application program being called as the server program.
Note: Only the first 8 characters are used.

Behaviour for protocol IMSLU62:
Name of the IMS transaction code to be invoked.
Note: Only the first 8 characters are used.

Behaviour for protocol SAPRFC:
Name of the SAP function module to be invoked.

Behaviour for protocol MAIL:
This is deprecated. Supported only for compatibility reasons. The alternative way to adress email is using *SetParam* with Parameter *MAIL-TO*.

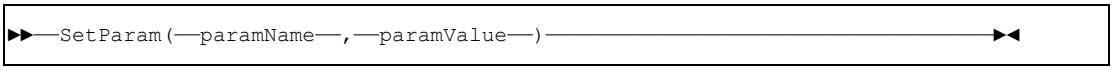


Figure 4-29: SetParam

The effect of **SetParam** depends on the used protocol. For more information see API call *nciSetParam*. The *paramName* and the *paramValue* has to be quoted.

Special behaviour for protocol SSL:

For a more detailed description see API call *nciSetParam* and the chapter “Protocol Specific Information”.

Values for ParamName	Description

SSL-PASSPHRASE	Passphrase for encrypted private key.
SSL-CACERTFILE	DER or PEM formated file containing all trusted CA certificates. Alternatively SSL-CACERTDIR can be used. Filename must contain absolute ore relative path information.
SSL-CACERTDIR	Directory containing all trusted CA certificates as separate files, using hash-values as file name. The hash value must be followedf by an additional '.0'. If a file with this value already exists,, this number has to be increased (e.g. 98bd731f.0 or 98bd731f.1). Alternatively SSL-CACERTFILE can be used.
SSL-CERTFILE	DER or PEM formated file containing own certificate / public key. This file can also contain the private key. Filename must contain absolute ore relative path information.
SSL-KEYFILE	DER or PEM formated file containing the private key. This file can also contain the certificate. Filename must contain absolute ore relative path information.
SSL-KEYFILETYPE	Filetype used for files in SSL-CACERTFILE, SSL-CACERTDIR, SSL-CERTFILE, SSL-KEYFILE and SSL-CRLFILE. All files have to be in the same format. Possible values are PEM (default) and DER.
SSL-KEYLABEL	Label of local/own certificate/private key to use for authentication. This parameter is only available for z/OS.
SSL-CRLFILE	DER or PEM formatted file containing one or more Certificate Revocation Lists (CRL) from trusted CAs. Filename must contain absolute ore relative path information.
SSL-SESSIONCACHE	N/A
SSL-PEERCERTCHECK	Select if communication partner (peer) must provide a certificate in order to establish a cominication. Possible values are:

NO	Peer certificate is not requested and not checked.
OPTIONAL	Peer certificate is not requested but checked if provided. Handshake failes if check failes.
REQUIRED (default)	Peer certificate is requested and checked. Handshake failes if check failes.

SSL-VERIFYDEPTH	Maximum allowed depth of Chain-of-Trust for certificate check. If Chain-of-trust exceeds this value, check failes. Default value 5.
SSL-CIPHERSPECS	Cipher specification to use. On Open System (Windows and Unix) this has be in the Syntax used by OpenSSL, on z/OS the number provided by SystemSSL are used.
SSL-RETRYTIMEOUT	Time between two retries, if SSL handshake stumbles because of IP timeout of nonblocking sockets.
SSL-MAXRETRIES	Number of retries before handshake finaly failes if it stumbles because of IP timeout of nonblocking sockets.
SSL-SECURITYFILE	Path and filename of configuration file with security (SSL) specific parameters in XML format.
SSL-EXPIRATIONCYCLETIME	Time between two cleanup runs to clear session cache from expired sessions.
SSL-SESSIONTIMEOUT	Validity time for SSL session before renegotiation is carried out.
SSL-ALLOWEXPIREDCERT	Temporarily allow certificates which are either not yet valid or already expired. If this parameter is set and a certificate is accepted due to this parameter, informational messages are written into the NCI log. Do not use this parameter if not absolutely necessary and reset it immediately after new certificates are provided.

Special behaviour for protocol MAIL:

Because the SMTP protocol relies on a smart usage, it is appropriate filling out the parameters “MAIL-SUBJECT” and “MAIL-FROM” correctly. This will avoid suspecting the email as spam.

Values for ParamName	Description for ParamValue
MAIL-SUBJECT	Specifies the subject of the email. It would be very friendly filling out this field.
MAIL-FROM	Sepcifies the sender of the email in complete email address format, e.g. cc.Middleware@t-systems.com. It would be very friendly filling out this field.
MAIL-REPLYTO	If the reply should send to a different email address, it is possible to declare this alternate address here. Please use this feature carefully.
MAIL-DOMAIN	The SMTP protocol communication relies on a correct established DNS infrastructure. If this requirement is not fulfilled it is possible to declare a DNS Domain name. This feature should used only by very experienced users.
MAIL-QUOTABLEPRINT	Specifies if special characters in the email should be quoted. Possible values are „ON“ and „OFF“. Default is „OFF“.
MAIL-TO	Specifies a receiver on the „To“ line. This can be called multiple times to send to multiple receivers.
MAIL-CC	Specifies a receiver on the „Cc“ line. This can be called multiple times to send to multiple Cc-receivers.
MAIL-RESET	Reset all parameters before another send.

Note: On z/OS it is absolutely required to set Data Conversion to “YES”, see *DataConv*.

Note: Email is an ASCII based protocol. So the user on non-ASCII systems like z/OS should take care of the correct special characters, e.g. the ‘@’ sign in email adresses like “ccMiddleware@t-systems.com” : on z/OS the used codepage is important, because the ‘@’ sign has different hexvalues in the different EBCDIC codepages, for example IBM-273 or IBM-1047. In this case the user has to verify that the codepage for the application and the editor or browser is the same.

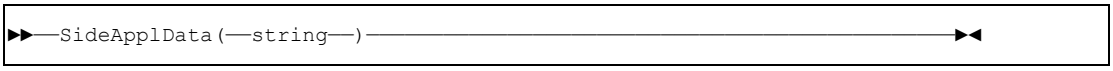


Figure 4-30: Sideinformation Application Data

The format and content of data depends on each application. This applicationspecific control parameters can be accessed by an application via function: *NCIGetSideAppldata*.

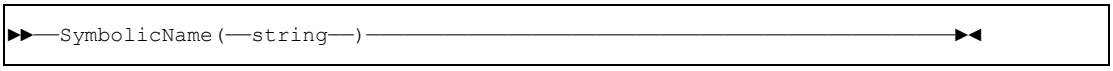


Figure 4-31: Symbolic Name

The *SymbolicName* will be used by an application if

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

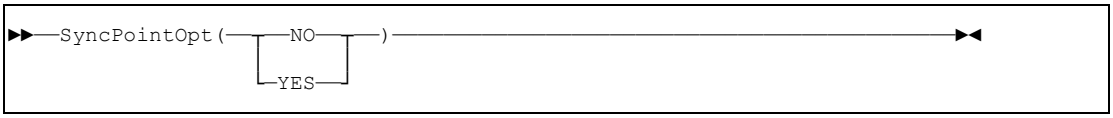


Figure 4-32: Syncpoint Option

Note: Syncpoint processing is only supported for protocol MQSeries and EXCI.

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*.

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.

Protocol EXCI

- NCIPutGet
  All PutGets belong to one LUW (Logical Unit of Work) until NCIBack is called to rollback the changes or NCICmit is called to commit all changes.
  Note: CICS Transaction Server 1.3 is required to use this feature.

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.

Protocol EXCI

- NCIPutGet
  Syncpoint is implicitly taken whenever the CICS program returns.

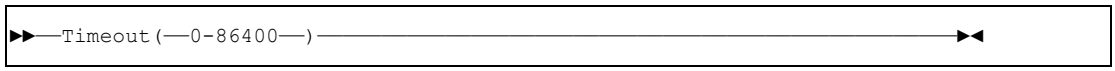


Figure 4-33: Timeout

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).

Behaviour for protocol MQSeries:

Value	Description
0	NCIPutGet: The application is blocked until a reply message will be received or until a connection error occurs. NCIGet: Return immediately if no suitable message can be recieved. The application does not wait for messages.
1-86400	NCIPutGet: 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). NCIGet: 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 NCIPut the timeout value is not supported.

Behaviour for protocol LU 6.2 and IMSLU62:

Value	Description
0	NCIPutGet: The application is blocked until a reply message will arrive or until a connection error will occur.
1-86400	NCIPutGet: 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).

Note: For the functions NCIPut and NCIGet the timeout value is not supported.

Behaviour for protocol TCP/IP:

Value	Description
0	NCIPutGet: The application is blocked until a reply message will arrive or until a connection error will occur. NCIGet: The application is blocked until the client sends data or the client closes the connection.
1-86400	NCIPutGet: 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). NCIGet:

The maximum time, expressed in seconds, to wait for data from the client or a connection termination.

Note: Timeout support for function NCIGet is currently only supported on OS/390.
Function NCIPUT does not support a timeout. *

Behaviour for protocol EXCI:
Not supported.

Behaviour for protocol SAPRFC:
Not supported.

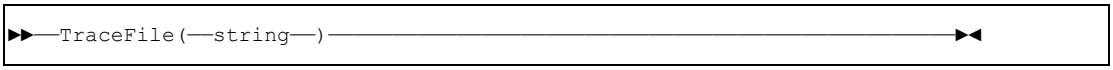


Figure 4-34: Trace File

Note: This keyword is not supported for operating system OS/390.

The default value for *TraceFile* is **ncitrace.log**.
Defines the name of the file, where NCI trace records will be written if tracing is in effect. Tracing can be activated by function call *NCISetTraceOpt* or via NCI sideinformation.

If tracing is enabled by defining a trace file at least the error messages are written to the tracefile, though TraceOpt(NONE) was set, i.e. independent of the settings of TraceOpt.

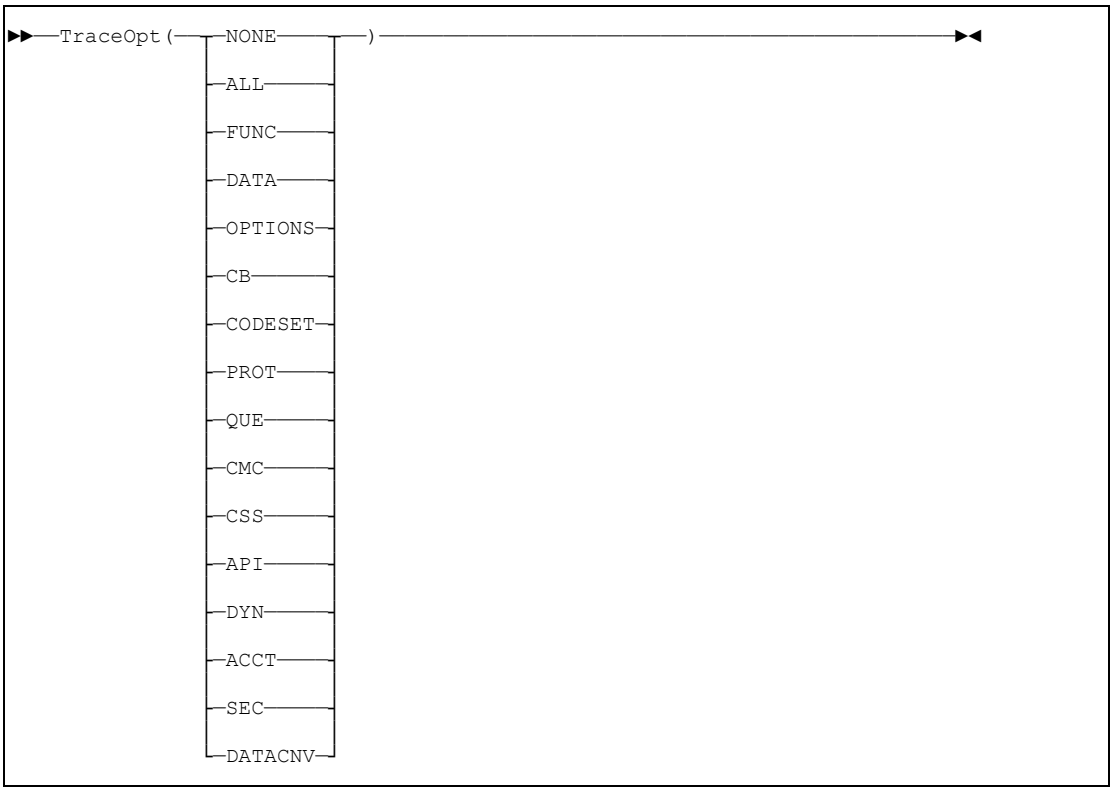


Figure 4-35: Trace Options

Note: Tracing is not supported on OS/390.

Note: Tracing options has changed slightly since NCI280.

If tracing is enabled by defining a trace file (see Parameter TraceFile) at least the error messages are written to the tracefile, though TraceOpt(NONE) was set, i.e. independent of the settings of TraceOpt.

Setup Trace Options.

Option	Description
NONE	Disable Trace.
ALL	Trace all.
FUNC	Function Trace.
DATA	Tracing of data flow
OPTIONS	NCI options being used.
CB	Tracing of NCI Control Blocks
CODESET	Tracing of Character Set handling
PROT	Specific transport protocol trace
QUE	Trace of request queuing
CMC	Trace of parallel server communication
CSS	Trace of parallel server handling
API	Trace of special API calls (e.g. RFC)
DYN	Trace dynamic loading
ACCT	Trace of accounting
SEC	Trace of security relevant settings

DATACNV Trace Data Conversion

Note: Tracing can produce a lot of output.

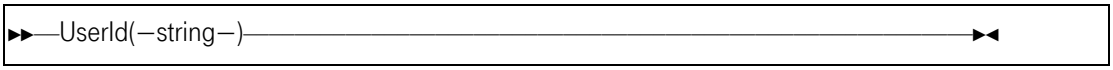


Figure 4-36: User Identifier

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 and LU 6.2: 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).

Note: Due to an extension for UserId since NCI310 to support UserId with more than 8 characters.

Index

Additional Features 107
AIX 15
AIX Format 17
ASCII 87, 91
Auto-Termination 63, 75
bff Format 17
CLIENT-ALIVECHECK 68, 79
CLIENT-ALIVECHECK-INTERVAL 68, 79
CILIENT-ALIVECHECK-CM 68, 79
Co nfiguration Parameter Nice-Value 69, 80
Communication Manager Unix 61
Communication Manager Windows 71
Configuration File 63, 75
Configuration Parameter Auto-Termination 63, 75
Configuration Parameter CLIENT-ALIVECHECK-CM 68, 79
Configuration Parameter Cycle-Time 68, 79
Configuration Parameter Default-TP 67, 78
Configuration Parameter Default-Userid 69, 80
Configuration Parameter Idle-Time 68, 80
Configuration Parameter IP-Port 63, 75
Configuration Parameter Msg-Log 65, 77
Configuration Parameter Pip Data 67, 78
Configuration Parameter Program-Name 67, 78
Configuration Parameter Root Access 66, 77
Configuration Parameter Server-Name 63, 75
Configuration Parameter Shadow-Passwd 66, 77
Configuration Parameter Shm-Key 64, 75
Configuration Parameter TEST-CONN 66, 78
Configuration Parameter TP-Max 67, 79
Configuration Parameter TP-Min 67, 79
Configuration Parameter Tp-Name 67, 78
Configuration Parameter Trace 64, 76
Configuration Parameter Trace-File 65, 77
Configuration Parameter Trusted-Host 65, 77
Configuration Unix 61
Configuration Windows (Win32) 71
Customization 61
Cycle-Time 68, 79
Data Representation 87
Default-TP 67, 78
Default-Userid 69, 80
depot Format 29
Digital Unix 45
EBCDIC 87, 91, 108
Gateway TP 107
HPUX 27
HPUX Format 29
Idle-Time 68, 80
Installation AIX 15
Installation Digital Unix 45
Installation HPUX 27
Installation LINUX 33
Installation OS/2 57
Installation SCO 51
Installation Sun Solaris 21
Installation Tandem/NSK 56
Installation Windows 39
Ip-Port 63, 75
Java 109, 110, 118, 133
LINUX 33
Linux Format 35
MSG-LOG 65, 77
NciHttpGateway 110
NCISTPGW 107
NICE-VALUE 69, 80
Options 83
OS/2 57
Parallel Server 61, 71
Pip-Data 67, 78
pkg Format 23
pkgadd 23
Program-Name 67, 78
Root-Access 66, 77
rpm 35
rpm Format 35
SCO 51
Security-Level 68, 80
Service TP's 107
Servlet 110
Servlets 109
Shadow-Passwd 66, 77
Shm-Key 64, 75
Sideinformation 83
smitty 17
Sun Solaris 21
Sun Solaris Format 23
swinstall 29
Tandem/NSK 56
TcbPrivilege 74
Test-Conn 66, 78
Tp-Max 67, 79
Tp-Min 67, 79
Tp-Name 67, 78
TP's 107
Trace 64, 76
Trace-File 65, 77
Trusted-Host 65, 77
Windows 39
Windows privilege 74
Windows System Service privileges 73
Windows System Service security 73
Windows System Service tool 74

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