Middleware Suite - Application Integration

NCI - Network Computing Interface

Installation and Customization for z/OS and OS/390
Implementation Guide
Version: 3.1

T · · Systems ·

Impressum

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

Document information

Name of file

Installation and Customization Guide for z/OS and OS/390

Version / Revision	Date	Revision
This edition relates to NCI version **NCI 3.1** • **PNCI310/QZ05046** on z/OS, OS/390 • **PNCI310/REL1003** on Unix/NT	30/01/2006 13:06:00	165

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

List of Figures

1 Migrating from a previous Release

1.1 What is new with NCI Version 3.1 and the latest PTF QZ05046

- SMF Recording
  NCI Communication Manager now supports the writing of a SMF record for every client request that was successfully served. The SMF record contains information like the client's userid, consumed CPU time, ... The intention of the SMF records is, to provide information for accounting purposes.
  NCI SMF recording can be enabled by setting the SMF-RECORDING(*type*,*subtype*) directive in the NCI parameter file. See chapter "4.5.9 SMF-RECORDING" for more information.

- New default code page
  Default code page on z/OS has been changed to IBM-273, the German EBCDIC code page. This was done, because almost all NCI Serveres use this code page.

- Code page support
  Code pages can be specified on client and server side to allow better data conversion between different code pages.

  On server side use:
  - the NCI API call nciSetLocalCodePage(...)
  - the LOCAL-CP(...) directive in the TP definition
  - the LOCALCP(...) directive in the NCI Side Info, if your server program uses the Side Info addressing.

  On client side use:
  - the NCI API call nciSetLocalCodePage(...)
  - the LOCALCP(...) directive in the NCI Side Info, if your client program uses the Side Info addressing.

- Data encryption enhancements
  - DES and Triple DES support.
  - Changed NCI API call: nciSetDataEncrypt(...) has been expanded to support the new encryption types.
  - New API call: nciSetCryptKey(...) to specify the encryption key to be used on client side.
  - On server side the NCI IP listener supports the new directive CRYPT-KEY(...) to specify the encryption key to be used by the listener.

- LU 6.2 usage enhancements
  - The default mode table entry used by NCI has been changed from LU62 to #INTER. #INTER is a LU 6.2 log mode that is provided by IBM as default. This eases the NCI LU 6.2 customization, because there is no need to add the old log mode LU62 to the VTAM mode table.
  - New NCI API call: nciSet62LogMode(...) to change the LU 6.2 log mode name to be used.
  - New NCI API call: nciSet62ReqLUPrefix (...) to change the prefix of LU 6.2 ACB names allocated by NCI clients.

- SSL Support (QZ04100 required)
  NCI now supports the communication via a SSL connection. On z/OS the NCI SSL

implementation is based on the z/OS System SSL feature. Both parts, the NCI client and the NCI server were enhanced to support SSL. Several new parameters and API calls were introduced to set the SSL specific properties.

- NCI Mail (QZ04100 required)
  New NCI addressing type 'MAIL'. E-mails can be easily sent using the standard NCI API. This enhancement provides the capability to directly send an e-mail out of a program. The frontend program NCIMAIL can be used as a standalone program to send e-mails via a batch job.

- NCI "Message Driven TPs" (QZ05046 required)
  NCI Communication Manager now supports MQSeries as a protocol for starting TPs. In order to use this feature, one or more MQ application queue listeners must be defined in the server configuration file

1.2 Migration steps

This chapter describes the steps to be taken, if you migrate from a previous NCI release.

NCI 3.1 was devopled with a great focus on compatibility to earlier NCI releases. Therefore there is only little effort required to migrate to NCI 3.1. There is no need to change any application code. Applications should run with NCI 3.1 without any modifications.

Steps to be taken:

- Install NCI Version 3.1
  See chapter "2 Installation" for more details.
  Note: NCI 3.1 no longer has any LPA modules. Please make sure, that all NCI 2.10 LPA modules are removed from LPA.

- Define new RACF profiles for dynamic system authorization
  The major name of the NCI Communication Manager load module has been changed from SCLDRV62 to NCICM. Therefore, if at least one NCI Communication Manager on the system uses "dynamic system authorization"[1], new RACF profiles are needed to reflect the name change.

  Check if you have defined profiles for the module SCLDRV62 in the RACF class $SEM.
  If yes, create a second set of those profiles with the new module name NCICM.
  Additionally add new profiles if you use different load libraries or DASD volumes for the new NCI version.
  Here is an example of a RACF command to add a profile for "dynamic system authorization" to the $SEM class :

```
rdefine $SEM nci-loadlibrary(NCICM)volser
UACC(READ)
```

  If the RACF class $SEM is RACLISTED, perform a RACF refresh on the $SEM class.

[1] *"Dynamic system authorization"* is a feature of the T-Systems software component SEM (System Enhancements and Modifications) that allows programs to dynamically gain a system authority like APF, SUPERVISOR state or KEYZERO, if the authorization is garnted by a corresponding profile in the RACF class $SEM.

Note: The NCI Communication Manager needs the APF system authorization to call authorized system services if one of the following NCI features is used:
- NCI communication Manager SMF recording is enabled.
- NCI communication Manager security is enabled. TPs are defined with either SECURITY-LEVEL(CHECK) or SECURITY-LEVEL(PROPAGATE).
- NCI communication Manager runs TPs that are defined with TP-INSTANCE(PROCESS)

- Activate new NCI version
  - Stop all running NCI applications.
  - Bring NCI version 3.1 into production.
  - Restart all NCI applications.

1.3 Known Problems/Incompatibilities

When you migrate from earlier NCI releases to NCI 3.1 you may encounter some problems that are well known but that could not be handeled programmatically by NCI. The steps to be taken then, depend on the single situation.

1.3.1 Codepage incompatibilities

In previous NCI releases the only way to change code pages, was to replace some certain NCI/SCL modules, or to rename translation modules that were provided by z/OS language environment. Therefore, if an application needed another code page than the one NCI used internally, "zapped" NCI modules, or renamed LE modules were placed in a separate load library in the STEPLIB concatenation. This method was very error prone and inconvenient. With NCI 3.1 there is no longer the need to do this. NCI 3.1 provides a code page support that allows the application to choose whatever code page is required. However the two methods are not really compatible.

In almost all cases the code page changes were done to activate the code page IBM-273. This is the German EBCDIC code page. Due to this fact, NCI 3.1 is delivered with IBM-273 as the default code page for data conversion.

Note: NCI z/OS applications that are using the IBM-273 code page should not encounter any compatibility problems. However applications that use another code page may encounter a problem. To solve this code page problem, those applications have to do one of the following application changes in order to set the correct code page:

On server side use:
- the NCI API call nciSetLocalCodePage(...)
- the LOCAL-CP(...) directive in the TP definition
- the LOCALCP(...) directive in the NCI Side Info, if your server program uses the Side Info addressing.

On client side use:
- the NCI API call nciSetLocalCodePage(...)
- the LOCALCP(...) directive in the NCI Side Info, if your client program uses the Side Info addressing.

2 Installation

2.1 Introduction

This chapter describes the installation of NCI on z/OS and OS/390. We recommend that you carry out the installation step by step as described in the following sub chapters.
The installation of NCI relies on SMP/E. It is delivered as a SMP/E function with the FMID PNCI310.

The NCI delivery package contains all data sets needed for the installation. Additionally the delivery also contains the latest product fixes that are available at the time of building the delivery package.

The SMP/E environment for NCI is necessary to support the data centres in case of problems.

Locations where SDM/zOS[2] is used should choose the SMP/E data set names according to the SDM naming conventions.
If the NCI installation is performed in the course of an SDM/zOS installation there is no need to do the installation. In this case the complete NCI SMP/E environment is already pre-installed in the SDM/zOS package. Furthermore the NCI elements were already copied from the SMP/E TARGET data sets to the SDM/zOS production libraries.
However, if you install SDM/zOS for the first time, you have to carry out the NCI customization described in chapter "3 Customization".

2.2 Prerequisites

The following software releases are prerequisites for the installation of NCI.

- OS/390 2.10 or later
- Language Environment
- SecureWay Communication Server (for IP Connectivity)
- Security Server (RACF) (2) or a functionally equivalent product
- MQSeries 1.2 or later (for MQSeries Connectivity)
- ISPF
- SMP/E

[2] *SDM – Software Delivery Mainframe* is a method for the employment of operation system software in the T-Systems Enterprise Services GmbH data centers. SDM is both, a standard multi-vendor package of z/OS respectivly OS/390 software and a method to implement this package on a system.

.

2.3 Installation

2.3.1 SMP/E Environment

FMID	PNCI310 (NCI 3.1)
Target zone	NCI31T
DLIB zone	NCI31D

2.3.2 SMP/E Files

Data Set Name	Zone	Contents
?hlqsmpe.NCI31.ANCISRC	DLIB	Sources (Dummy - not used)
?hlqsmpe.NCI31.ANCILOAD	DLIB	Linklib modules
?hlqsmpe.NCI31.ANCIMAC	DLIB	Macros
?hlqsmpe.NCI31.ANCIH	DLIB	C Headers
?hlqsmpe.NCI31.ANCISAMP	DLIB	Samples
?hlqsmpe.NCI31.ANCIEXEC	DLIB	REXX EXECs
?hlqsmpe.NCI31.ANCIPNL	DLIB	ISPF Panels
?hlqsmpe.NCI31.ANCIMSG	DLIB	ISPF Messages
?hlqsmpe.NCI31.ANCISKL	DLIB	ISPF Skeletons
?hlqsmpe.NCI31.SNCISRC	TARGET	Sources (Dummy - not used)
?hlqsmpe.NCI31.SNCILOAD	TARGET	Linklib modules
?hlqsmpe.NCI31.SNCIMAC	TARGET	Macros
?hlqsmpe.NCI31.SNCIH	TARGET	C Headers
?hlqsmpe.NCI31.SNCISAMP	TARGET	Samples
?hlqsmpe.NCI31.SNCIEXEC	TARGET	REXX EXECs
?hlqsmpe.NCI31.SNCIPNL	TARGET	ISPF Panels

?hlqsmpe.NCI31.SNCIMSG	TARGET	ISPF Messages
?hlqsmpe.NCI31.SNCISKL	TARGET	ISPF Skeletons
?hlqsmpe.NCI31.SNCIHFS	TARGET	HFS elements
?hlqsmpe.NCI31.CSI ?hlqsmpe.NCI31.SMPLOG ?hlqsmpe.NCI31.SMPMTS ?hlqsmpe.NCI31.SMPSCDS ?hlqsmpe.NCI31.SMPSTS		SMP/E data sets

Figure 2-1: SMP/E data sets

2.3.3 Installation Files

Usually NCI is delivered on a cassette containing the files listed in the table below. The files F0-Fn are structured in the SMP/E TXLIB format and contain the entire product (FMID).

File #	DSNAME	VOLSER	Description
1	SYS1.PNCI310.INSTLIB	NCI310	Installation library
2	SYS1.PNCI310.F0	NCI310	SMP/E MCS Statements
3	SYS1.PNCI310.F1	NCI310	Samples
4	SYS1.PNCI310.F2	NCI310	ASM macros
5	SYS1.PNCI310.F4	NCI310	C header files
6	SYS1.PNCI310.F5	NCI310	REXX execs
7	SYS1.PNCI310.F6	NCI310	ISPF panels
8	SYS1.PNCI310.F7	NCI310	ISPF messages
9	SYS1.PNCI310.F8	NCI310	ISPF skeletons
10	SYS1.PNCI310.F9	NCI310	Modules

Figure 2-2: Installation files

2.3.4 Restoring the Installation Library from Tape

This chapter is only necessary if you received the NCI installation on a tape or cassette.
The first thing to do is to load the installation library from the tape. The installation library contains all the jobs that are required for the further installation of NCI.

```
//TAPEINST JOB
 //*****************************************************************
 //*                                                               *
 //*        JOB TO LOAD THE NCI VERSION 2.10.0 INSTLIB             *
 //*        LIBRARY FROM TAPE                                      *
 //*                                                               *
 //*---------------------------------------------------------------*
 //* REQUIRED CHANGES:                                             *
 //*                                                               *
 //* - CHANGE ?UNIT     TO THE UNIT NAME OF YOUR TAPE DRIVES       *
 //* - CHANGE ?HLQ      TO THE HIGH LEVEL QUALIFIER FOR YOUR       *
 //*                    INSTLIB                                    *
 //*****************************************************************
 //*
 //COPY     EXEC PGM=IEBCOPY
 //SYSPRINT DD SYSOUT=*
 //SYSUT1   DD DISP=OLD,DSN=SYS1.PNCI310.INSTLIB,UNIT=?UNIT,
 //         VOL=(PRIVATE,RETAIN,SER=NCI310),LABEL=(1,SL,EXPDT=98000)
 //SYSUT2   DD DSN=?HLQ.PNCI310.INSTLIB,
 //         DISP=(NEW,CATLG),UNIT=SYSALLDA,SPACE=(TRK,(81,20,20)),
 //         BLKSIZE=27920,LRECL=80,RECFM=FB,DSORG=PO
 //SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),DISP=(,DELETE)
 //SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),DISP=(,DELETE)
 //SYSIN    DD *
   COPY  INDD=SYSUT1,OUTDD=SYSUT2
 //
```

Figure 2-3: Restoring the Installation Library

2.3.5 Installation – Step by Step

The INSTLIB contains the installation jobs INST01 to INSTnn. The jobs have to be executed in the numbered order. To adapt the jobs to your needs, please follow the instructions contained in each job.

NOTE:
The jobs suffixed with a 'H' are only required, if you intend to install the following NCI features:

- NCI Web PlugIn for the IBM HTTP Server for z/OS.
- NCI Java classes.

These features will be installed into HFS. However they are not needed to use the NCI base functions. To make the installation of NCI easier, these features are not part of the NCI base FMID. They are shipped in a separate PTF.

In the following installation it is assumed that the **NCI TARGET** zone and **NCI DLIB** zone will be connected to an existing SMP/E **GLOBAL** zone.

Job	Description

INST01	Load NCI installation files from the tape cartridge.
INST02	Define NCI SMP/E TARGET and Dlib CSI's
INST03	Initialize the NCI SMP/E CSI's
INST04	Connect NCI SMP/E CSI's to the GLOBAL CSI
INST05	Allocate NCI SMP/E data sets
INST05H	Optional: Allocate NCI SMP/E data sets for NCI HFS elements
INST06	Add SMP/E DDDEF entries to NCI TARGET and DLIB zone
INST06H	Optional: Add SMP/E DDDEF entries to NCI TARGET and DLIB zone for NCI HFS elements
INST07H	Optional: Create the mountpoint for the NCI target HFS and mount the allocated HFS to this moutpoint. To permanently mount the NCI HFS, add the mount statement to the BPXPRMxx parmlib member.
INST08	Receive the NCI FMID
INST09	Apply the NCI FMID
INST10	Receive the NCI PTFs shipped with this delivery
INST10H	Optional: Receive the PTF for the NCI HFS elements
INST11	Apply all NCI PTFs. **Note:** Some PTFs may contain a ++ HOLD section. Read this section carefully and take the described actions.
INST12	Accept NCI FMID and all PTFs
INST13	Copy job to copy NCI elements from the SMP/E TARGET data sets to the production environment
INST13H	Optional: Copy job to copy the NCI SMP/E target HFS to a production HFS.

Figure 2-4: Installation Jobs

Now you have successfully finished the NCI installation process. Before you can use NCI some customization is necessary. Please refer to the customization chapter

3 Customization

Before NCI can be used on your system, there is some customization necessary. This chapter only describes the basic customization. To setup all the other NCI features, please refer to the next chapters.

3.1 VTAM definitions for NCI clients

If you intend to use NCI clients that use the LU 6.2 protocol, a pool of NCI VTAM LU-Names must be defined that are used by the NCI clients to establish a LU 6.2 connection. This pool of LU-Names is shared among all NCI clients on the system. When a NCI client tries to open a LU it uses ACB names in a predefined range. By default ACB names for LU 6.2 LUs have to be prefixed with “SCL6”. The remaining characters (up to 8) must be numeric and in the range from 0000 to 9999.
It is also possible to define additional pools of client LUs. To use another pool than the default, the application program has to use the NCI API call *NciSet62ReqLUPrefix()* to change the pool prefix.

The number of needed LU-Names (ACBs) strongly depends on the number of concurrently working NCI clients that use the LU 6.2 protocol. It is recommended to define at least 100 LU-Names. If all LU-Names are in use, the following error message will occur.

```
NCI1112E … ALL NCI REQUESTOR ACBS ARE IN USE. LAST CHECKED ACB <ACB name>.
```

The “Figure 3-1: VTAM definition for NCI LU 6.2 clients” shows a sample VTAM definition that is available in the sample member NCIVTAMR. You may have to adapt the LU names to the naming conventions of your installation. However, as already mentioned the ACB names must not be changed. The used VTAM log mode is an IBM default that is shipped with VTAM and should be available in all locations.

Note: The definitions for LU 2 connections are for NCI downward compatibility and must not be removed.

```
AMX1NCI VBUILD TYPE=APPL
*              STATOPT='NCI CLIENTS'
*
*   DOC: THIS MEMBER CONTAINS VTAM APPLICATION DEFINITIONS FOR
*        NCI CLIENTS.
*        SCL00000 - SCL00099 (LU TYPE 2)
*        SCL60000 - SCL60999 (LU TYPE 6.2)
*
************************************************************
* ACB'S FOR NCI CLIENTS   LU-TYPE 2
************************************************************
APX1SC?? APPL  ACBNAME=SCL000??,
               AUTH=(ACQ,NOPASS)
*********************************************************************
* ACB'S FOR NCI CLIENTS   LU-TYPE 6.2
*********************************************************************
APX1S??? APPL  ACBNAME=SCL60???,
               APPC=YES,
               DLOGMOD=#INTER,
               PARSESS=YES,
               SECACPT=NONE,
               VPACING=1
```

Figure 3-1: VTAM definition for NCI LU 6.2 clients

3.2 Preparing CICS for NCI

If you intend to use NCI within CICS, the NCI modules have to be defined to the CICS system. The sample member (NCICICS) contains the required CICS resource definitions (programs and tasks).
The resource group should be included in the CICS start p group list. Change the list name accordingly.
Update the CSD with the CICS offline utility program, DFHCSDUP, using this sample member as input.
The CEDA transaction can be used to install the resource group dynamically.
For details about these utilities refer to the appropriate CICS manuals provided by IBM.

3.3 Preparing IDMS for NCI

If you intend to use NCI within IDMS, a IDMS system generation for NCI is necessary.
A sample member (NCISGIDM) is provided that should be used as input to an IDMS system generation

4 NCI Communication Manager

4.1 Generally

NCI communication manager is a very flexible and simple to use application server. It allows hosting of server applications written in different languages like Assembler, Cobol or C/C++. Both, the client and server programs have to use the NCI API to establish the communication. Because of the easy use of the NCI API, it is very easy to develop complex client server applications.

NCI CM supports the transport protocols LU 6.2, TCP/IP, SSL and MQSeries. NCI CM is a “multi protocol listener”, that means it can handle different protocols concurrently. However from the application point of view there is no need to know all the specific details about the used protocol. All the protocol handling stuff is shielded by the NCI API.

Additionally NCI CM provides a powerful Thread/Process and Security management. Server application instances (throughout this document often referred to as TPs) can run as threads (subtasks) or as separate processes (address spaces). The security management allows to verify userids and to propagate the client userid to the server instance.

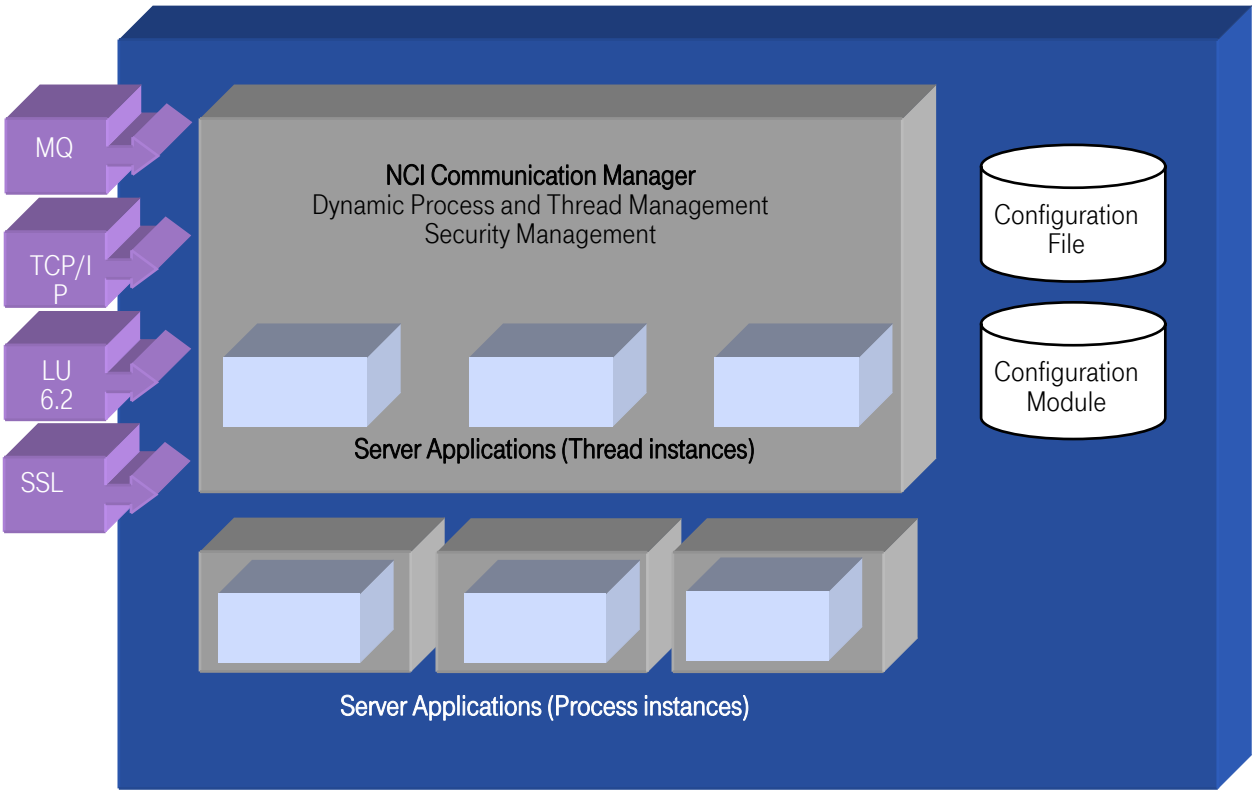

Figure 4-1: NCI Communication Manager Overview

4.2 Customizing NCI Communication Manager for z/OS and OS/390

NCI Communication Manager for z/OS and OS/390 is a typical instance software. That means that you can run multiple instances of a NCI Communication Manager on the same system. However each

instance needs its own configuration. The customization described in the next chapters has to be done for each instance.

On z/OS and OS/390 the configuration of a NCI Communication Manager is divided into 2 parts. One part that contains the data center specific parameters (configuration file) and a second part that specifies the application specific parameters (configuration module). The splitting was done because of the different responsibilities. In a production environment we usually have people that are responsible for the data center and others that are responsible for the application. This separation makes it possible that the application developers provide the configuration module together with their application.

4.2.1 Configuration file

A configuration file contains the data center specific configuration parameters such as the VTAM LU-Name or TCP/IP port number of the NCI communication manager.
The configuration file is addressed from the NCI Communication Managers start JCL via the DD-name **NCIPARAM**. The Record Format has to be FB and the Record Length can vary from 80 to 255.

"Figure 4-2: Sample configuration file for NCI Communication Manager on z/OS" shows a sample configuration. The configuration parameters are described in detail in chapter "4.5 NCI CM configuration file – Parameter Reference".

```
NCIPARAM  DD   *
*
* SAMPLE NCI configuration file
*
       SERVER-NAME(MYSERVER)
       MODULE-NAME(NCICONF)
*
       VTAM-APPLID(APX1APP1)
*
       IP-PORT(3490,53.113.127.10)
         MAX-SOCK(500)
         CRYPT-KEY(89123456A3B5C6B7)
*
       IP-PORT(3491)
*
       SSL-PORT(3453)
         Set-Param(SSL-KeyFileType,SAF)
         Set-Param(SSL-KeyFile,NCISRV1_RING)
         Set-Param(SSL-CipherSpecs,0A2F)
*
       MQ-QUEUE(SAZ1.NCI.INITQ,MQX1)
*
       MQ-APPL-QUEUE(SAZ1.NCI.APPLQ,MQX1)
*
       TRUSTED-HOST(53.1.9.10)
       TRUSTED-HOST(53.1.9.11)
*
       TP1-PROCNAME(TP1JCL)
       TP2-PROCNAME(TP2JCL)
*
       TRACE(NONE)
```

Figure 4-2: Sample configuration file for NCI Communication Manager on z/OS

4.2.2 Configuration module

The configuration module contains the application specific configuration parameters such as TP names, program names, security settings and so on. These parameters control the behaviour of an application.
The configuration module is addressed via the parameter **MODULE-NAME(...)** in the configuration file.

In most cases the configuration module is provided by the application developers together with the application. The module has to be created out of a readable source file using the NCI ISPF dialog.

“Figure 4-3: Sample configuration module for NCI Communication Manager on z/OS” shows a sample source of a NCI configuration module. See chapter “4.6.2

NCI CM Configuration Module" for more details.

```
*************************************************************
* Source NCI Configuration Module for APPLICATION: MYAPP
*************************************************************
      AUTO-TERMINATION(NO)
*
*
      TP-NAME(TSTTP1)
        DEFAULT-TP(NO)
        PROGRAM-NAME(NCITSTCS)
        PIP-DATA(PARM1,PARM2)
        TP-MIN(1)
        TP-MAX(10)
        CYCLE-TIME(0)
        IDLE-TIME(10)
        SECURITY-LEVEL(NONE
        LOCAL-CP($IBM273)
        TP-INSTANCE(TASK)
*
      TP-NAME(TSTTP2)
        PROGRAM-NAME(NCITSTCS)
        TP-MIN(0)
        TP-MAX(1)
        SRV-TIMEOUT(10)
*
```

Figure 4-3: Sample configuration module for NCI Communication Manager on z/OS

4.2.3 Security Definitions

To be able to use the following listed features the NCI Communication Manager needs APF authorization.

- NCI communication Manager SMF recording is enabled.
- NCI communication Manager security is enabled. TPs are defined with either SECURITY-LEVEL(CHECK) or SECURITY-LEVEL(PROPAGATE).
- NCI communication Manager runs TPs that are defined with TP-INSTANCE(PROCESS)

To gain the APF authorization either the NCI load library must be APF authorized, or you must allow NCI to gain the APF authorization dynamically via the "Dynamic System Authorization"[3] .
Please take into account, if you run the NCI Communication Manager from APF authorized libraries, the load libraries that contain the user application code must also be APF authorized. This is because the operating system denies the loading of non APF authorized code into an APF authorized process (Abend S306).
Running the application code from APF authorized libraries is a potential security risk. Only do this, if you can fully trust the application. In all other cases we recommend a dynamic authorization.

To allow the NCI Communication Manager code to dynamically authorize itself, the T-Systems software component SEM[4] must be installed and customized on your system.
Additionally a profile in the RACF class $SEM is required. Use the following RACF command to add a profile to the $SEM class. This profile controls the usage of the dynamic aurhorization for the NCI Communication Manager code.

[3] *"Dynamic system authorization"* is a feature of the T-Systems software component SEM (System Enhancements and Modifications) that allows programs to dynamically gain a system authority like APF, SUPERVISOR state or KEYZERO, if the authorization is garnted by a corresponding profile in the RACF class $SEM.

[4] SEM (System Enhancements and Modifications) is a T-Systems software component that provides a lot of enhancements to the z/OS operating system.

```
rdefine $SEM nci-loadlibrary(NCICM)volser UACC(READ)
```

4.3 Starting the NCI Communication Manager

The NCI Communication Manager can be started in 2 ways.

Either start it as a batch job or define a Started Task. Use the members

- NCICM
  Start NCI Communication Manager as a batch job
- NCIPROC
  Start NCI Communication Manager as a Started Task.

in the SNCISAMP as sample.

```
//NCICM    JOB
//NCICM    EXEC PGM=NCICM
//STEPLIB  DD  DISP=SHR,DSN=nci loadlib
//         DD  DISP=SHR,DSN=application loadlib
//NCIPARAM DD   *
*
*     NAME OF THE SERVER
       SERVER-NAME(MYAPPSRV)
*
*     NAME OF NCI CONFIGURATION MODULE FOR NCITST APPLICATION
       MODULE-NAME(MYAPP
*
*     VTAM APPLICATION NAME IF SERVER SHOULD SUPPORT LU 6.2
       VTAM-APPLID(APX1STST)
*
*     TCP/IP PORTNUMBER IF SERVER SHOULD SUPPORT TCP/IP
*      IP-PORT(3450)
```

Figure 4-4: Starting NCI Communication Manager as Batch Job

Note: If a NCI server application should be started in a separate address space (*TP-INSTANCE(PROCESS)*)refer to "4.6.2.11

TP-INSTANCE" the NCI Communication Manager must be started as a Started Task.

To stop a running NCI Communication Manager use the operator stop command.

```
f jobname,STOP
```

4.3.1 Operator Interface

At startup the NCI Communication Manager automatically establishes an operator interface.

The following operator commands are supported:

Command	Description
F jobname,HELP	Display available commands.
F jobname,STOP	Stop all TP's and termiante the Communication Manager.
F jobname,STOP=tpname	Stop a specific TP. All TP instances of this TP will be stopped. New client requests for this TP will be rejected. However client request contained in the TP request queue are finally processed. Therefore if there are many client requests in the TP's request queue the stop will be delayed.
F jobname,START=tpname	Release a specific TP previously stopped with the STOP=tpname command. This command does not start any TP instances, the TP will be only enabled to accept new client requests.

Figure 4-5: NCI Communication Manager Operator Interface

4.4 Using the NCI Communication Manager via Call Interface

This chapter is not of common interrest. It's intended for those users who want to start the NCI Communication Manager from within their own code as a subtask or subroutine.
The common way to start NCI Communication Manager was already described in the previous chapter. However for special purposes it is required to start the NCI CM directly from the user code. A good example is an application were the communication is only a small part of the functionality. To satisfy these needs NCI CM provides a call interface. The call interface can be invoked using the MVS LINK or ATTACH assembler macros.

Format of Call Parameter:

Note: Register 1 must be set to zero if no parameters should be passed. Otherwise Register 1 must point to the following parameter-list:

Parameter	Number of Bytes	Description
Parameter 1 (required)	4	The address of an ECB used for communication between NCI CM and the task that invoked the communication manager. If this ECB is posted with the postcode **NCIORI_PCODE_OPER** (defined in macro **NCIORI**) NCI CM assumes that an operator request is passed via the NCI CM operator request interface. In all other cases NCICM interprets the post as a stop request and terminates.

		If this parameter is set, NCI CM doesn't open the operator interface described in chapter "4.3.1 Operator Interface" , because NCI CM assumes that the invoking task has already build an operator interface. In this case the only way to request operator commands from NCI CM is the NCI operator interface described in parameter 3. **Note:** The parameter must be set to zero if it is not used.

Parameter 2 (required)	4	The address of a buffer containing the NCI CM initial parameters. All keywords which are valid for the NCI Configuration file may also be specified through this buffer. However if this parameter is set, the NCI configuration file (DDNAME: NCIPARAM) will be ignored.
		The passed buffer must have the following layout:
		
		`PARMBUFF DS 0F`
		`DS Xl1 Length of Keyword-name`
		`DC CL_ Keyword-name`
		`DC XL1 Length of Keyword-value`
		`DC CL_ Keyword-value`
		`...`
		`...`
		`DC xl1'00' End of List indicator`
		
		If this parameter is not used it must be set to zero, if it is the last one in the list, the high order bit must be set to indicate the end of the parameterlist.

Parameter 3 (optional)	4	The address of the NCI operator request interface block. The interface is mapped by the macro **NCIORI** and may be used to pass operator commands from a task owning the operator interface to NCI CM. To use the operator interface you must supply the NCI specific part of the operator command in the interface block and then post the ECB specified in parameter 1 with the postcode **NCIORI_PCODE_OPER**

Figure 4-6: NCI Communication Manager Call Interface

4.5 NCI CM configuration file – Parameter Reference

In the following chapters the NCI CM configuration file parameters are described in detail. The member NCIPARAM in the NCI samples data set SNCISAMP is a sample configuration file with a lot of commet text.

4.5.1 DUB-DEFAULT-OPTION

Specify the default DUB settings, for dubbing MVS work to a z/OS USS process/thread.

Syntax:

```
DUB-DEFAULT-
OPTION(DUBTHREAD/DUBPROCESS/DUBTASKACEE/DUBPROCESSDEFER)
```

DUBTHREAD
Dub as thread

DUBPROCESS
Dub as a process

DUBTASKACEE
Dub as task ACEE

DUBPROCESSDEFER
Dub as process but defer dubbing

Default: n/a

For a complete description refer to: “OS/390 Unix System Services Programming: Assembler Callable Services”.

Note:
DUBPROCESS, is required, if a NCI TP program must run in a POSIX environment (LE runtime option POSIX(ON)). For example, if you want to use the SAPRFC communication within a NCI TP you have to set
DUB-DEFAULT-OPTION to **DUBPROCESS**.

4.5.2 IP-PORT

Use this parameter group to define an NCI CM TCP/IP listener.
This parameter can be used multiple times to support more than one port. That means, NCI CM is able to listen on more than one TCP/IP port.

Syntax:

```
IP-PORT(portNumber,ipAddress, additional parameters)
  MAX-SOCK(maxSocketNumber)
  CRYPT-KEY(cryptKey)
..SET-ENV(envString)
```

portNumber char(1-60)
TCP/IP service name or port number to be opened to accept client requests.

ipAddress char(1-256), optional
TCP/IP address or host name to bind this port only to this specific IP address **.**

additional parameters char(100), optional
Additional parameters for this listener. This parameter is for compatibility with earlier NCI releases and should no longer be used. The only supported parameter in this place is MaxSock(...). We recommend to use the extra MAX-SOCK(...) parameter now.

maxSocketNumber (45 – 2000), optional
Number of max. open sockets supported by this TCP/IP listener.
Default: 45

cryptKey char (16-80), optional
Sets the user data encryption key used to decrypt/encrypt data that this listener transfers.
Clients that want to communicate with this listener must use the same key. Each byte of the key must be defined as 2 hexadecimal characters. The entire length of the required key depends on the used encryption algorithm.

DES	length 8 bytes	string length 16 characters
3DES	length 24 bytes	string length 48 characters
5DES	length 40 bytes	string length 80 characters

envString char (1-256), optional
Parameter to set environment variables for this listener in the form ***name=value***.
This parameter can be repeated multiple times, to set more than one environment variable.
Example:

```
  SET-ENV(_TZ=MET-1DST,M3.5.0,M10.5.0)
```

4.5.3 MODULE-NAME

Syntax:

MODULE-NAME(***moduleName***)

moduleName char(1-8)
Name of the application specific NCI configuration module to be used by the server. This module defines the TPs that are provided by the application. See chapter “” for more information.

4.5.4 MQ-QUEUE

Use this parameter to define an NCI CM MQSeries trigger monitor.
This parameter can be used multiple times to support more than one triggered queue.

Syntax:

`MQ-QUEUE(`***`queueName,queueManagerName`***`)`

`queueName` char (1-48)
Specifies the MQSeries initiation queue name.

`queueManagerName` char (1-48)
Name of the corresponding queueManager.

4.5.5 MQ-APPL-QUEUE

Use this parameter to define an NCI MQSeries listener to support “NCI message-triggered TPs”. This parameter can be used multiple times to specify more than one queue.

Syntax:

MQ-APPL-QUEUE(***queueName*,*queueManagerName***)

queueName char (1-48)
Specifies the name of the MQSeries application queue to receive messages intended for TPs.

queueManagerName char (1-48)
Name of the corresponding queueManager.

4.5.6 RESET-APF

Reset an existing APF authorisation.
If the NCI Communication Manager is loaded from an APF authorized library the whole address space gets APF authorized. This leads to errors (abend S306) if NCI CM tries to load user application modules from NON APF authorized libraries. RESET-APF(YES) can be used to reset this initial APF authorization.

Syntax:

RESET - APF (**YES**)

Default: NO

4.5.7 RESTRICT-CLIENTACCESS

Syntax:

`RESTRICT-CLIENTACCESS(`***`Y/N`***`)`

Y/N
If RESTRICT-CLIENTACCESS(Y) is set to 'Y' the NCI Communication Manager performs additional security checks to insure, that the requesting client is authorized to access a specific NCI transaction program (TP).
With this parameter set to 'Y' the client userid must have READ permission to the profile

`NCI.CLIENTACCESS.`*`servername.tpname`*

in the FACILITY class.

Where:
`servername` is taken from the configuration parameter SERVER-NAME and
`tpname` is the name of the requested TP.

Note: Setting RESTRICT-CLIENTACCESS(Y) does not mean that NCI CM also verifies the userid provided by the client. The userid is taken as is. To do userid verification the TP must be specified with SECURITY-LEVEL(CHECK | PROPAGATE)

Default value: N

4.5.8 SERVER-NAME

Syntax:

SERVER-NAME(***serverName***)

serverName (char 1-8)
Name of the NCI server. The defined name is used, in several ways.

- In conjunction with the defined MODULE-NAME, to build internally used resource names
- To create entity names to check against “restricted client access”.

Note: It is strongly recommended to start each NCI Communication Manager on the same system with a different name. Under some circumstances NCI CM startup fails, if another NCI CM with the same name is already running on the same system.

Default value: $DEFAULT

4.5.9 SMF-RECORDING

Enable NCI SMFrecording. If SMF recording is enabled, NCI writes a SMF record for every processed client request. These records can be used for accounting purposes. The layout of the NCI SMF record can be found in the macro NCIDSSMF contained in the data set SNCIMAC.

Syntax:

SMF-RECORDING(***SMF record type,record sub type***)

SMF record type number(128-255)
SMF record type. Choose a type that is currently not used in your location.

SMF record sub type number(1-255)
Sub type. Choose any number that fits for your needs.

Note: NCI does not require a particular number. However the choosen numbers for type and sub type should suit your SMF processing.

Default: n/a

4.5.10 SSL-PORT

Use this parameter group to define an NCI CM SSL listener. The parameter group can be specified multiple times to define multiple SSL listeners. The NCI SSL implementation on z/OS is based on z/OS system SSL and supports SSL V3.0 and TLS V1.0. Keyrings can be setup as HFS key databases or in RACF. Also refer to chapter "6-Setting up NCI for SS" for more information.

Special API calls were introduced that can be used within an application program to access SSL information after a SSL session was established. For more information, refer to the "*NCI Application Programming Reference*" documentation.

Syntax:

```
SSL-PORT(portNumber,ipAddress)
  SET-PARAM(paramName,paramValue)
  MAX-THREADS(maxThreads)
  MIN-THREADS(minThreads)
  MAX-WORKQUEUEDEPTH(maxWorkQueueDepth)
..SET-ENV(envString)
```

portNumber char(1-60)
TCP/IP service name or port number to be opened to accept SSL client requests.

ipAddress char(1-256), optional
TCP/IP address or host name to bind this port only to this specific IP address **.**

paramName,paramValue char(1-30),char(1-1024)
Use name value pairs to define a set of properties for this SSL listener.
The following SSL parameters are currently supported:

- SSL-KEYFILETYPE
  Type of key store
  - SAF Key database is a is a RACF keyring
  - GSKDB Key database file in the HFS.
  Default: GSKDB

- SSL-KEYFILE
  Name of the key database file or RACF key ring. If a key database file is specified, it mustbe an existing z/OS HFS file. If a RACF key ring is specified, it must be an existing RACF keyring and the server user ID must have READ access to the IRR.DIGTCERT.LISTRING and IRR.DIGTCERT.LIST resources in the FACILITY class.
  Default: n/a

- SSL-PASSPHRASE
  Key database file password
  Note: This parameter has to be omitted when a RACF keyring is used or when a password stash file is used.
  Default: n/a

- SSL-STASHFILE
  Name of a file that contains the password for the key database file
  Note: This parameter must be omitted when a RACF keyring is used.
  Default: n/a

- SSL-KEYLABEL
  Label of the key that should be used. A RACF keyring or also a key database in the HFS can hold more than one key. To select a specific key, the label of that key can be specified with this parameter. If this parameter is omitted, the default key in the keyring/key database will be used.
  Default: n/a

- SSL-CIPHERSPECS
  List of SSL V3.0 and TLS V1.0 cipher specs that this SSL listener accepts. The cipher list can be combined in any order, however the position of a cipher spec in the list defines its usage preference (first one is prefered).
  The cipher specs supported depend on the used version of z/OS System SSL. Newer z/OS versions support more ciphers. Please refer to the z/OS System SSL documentation of your z/OS release to get a full list of supported ciphers. z/OS 1.4 supports the following ciphers:
  01 = NULL MD5
  02 = NULL SHA
  03 = RC4 MD5 Export
  04 = RC4 MD5 US
  05 = RC4 SHA US
  06 = RC2 MD5 Export
  09 = DES SHA Export
  0A = Triple DES SHA US
  2F = 128-bit AES encryption with SHA-1
  35 = 256-bit AES encryption with SHA-1
  Note: The cipher that is used for a secure SSL connection is negotiated by the client and the server during the SSL handshake. A SSL handshake is done whenever a client opens a new SSL connection to the server. The first cipher spec found in both cipher lists is used for that connection. Therefore, at least one cipher must be accepted by both sides, otherwise the SSL connection won't be established. This mechanism can be used by servers, to force clients to use a specific cipher spec. For example setting the cipher spec to '0A2F35' means, that only clients that support strong cipher algorithms like Triple DES, AES 128 or AES 256 can connect to the server.
  Default: 05040A0306090201

- SSL-PEERCERTCHECK
  The following fixed constants can be set for this parameter
  REQUIRED
  Perform the SSL handshake as a server that requires client authentication. The client certifiacte is validated against the server's keyring. The the SSL handshake fails, if the client certificate is not valid.
  OPTIONAL
  The client can optionally send a certificate. If a client certificate is available, it will be validated against the server's keyring. The handshake fails, if the client certificate is not valid.
  NO
  Client certificate not requested and not validated.
  Default: REQUIRED

- SSL-SECURITYFILE
  File name and section name of the NCI SSL security file.
  Besides the possibility to define the SSL listener parameters directly in the NCI CM configuration file, it is also possible to provide the SSL parameters via a XML based NCI security file (see sample NCISECXL in the SNCISAMP library).
  The NCI security file can have multiple named sections, each section holding a different SSL definition. The security file must be speciefied as filename:section, to address the file and section within the file. The filename can be a:

HFS file /var/ssl/securityFile.xml:Sample_1
MVS data set name SYS0.SSL.SECURITY(FILE1):Sample_1
DD name DD:SECURITY:Sample_1
Note:
(1) SSL listener parameters defined via SET-PARAM(...) in the NCI CM configuration file, take precedence over the same parameters defined in the security file.
(2) To use a security file, the z/OS XML 1.6 feature is required. The XML loadlib must be part of the LINKLIST or has to be defined in the NCI CM STEPLIB.
Default: n/a

maxThreads numeric (1-99), optional
Maximum number of SSL helper threads that should be started by this SSL listener. Every open SSL connection occupies one helper thread. If all helper threads are busy, new SSL connection requests will be queued up to the limit defined in `MAX-WORKQUEUEDEPTH`.
Default: 20

minThreads numeric (1-99), optional
Number of SSL helper threads that are prestarted during the startup of the SSL listener main thread.
Default: 0

maxWorkQueueDepth numeric (1-99), optional
Maximum number of SSL connections that will be queued in the NCI internal work queue, if all SSL helper threads are busy.
Default: 10

envString char (1-256), optional
Set environment variable for this listener in the form ***name=value***.
This parameter can be repeated multiple times, to set more than one environment variable.
Example:

```
SET-ENV(_TZ=MET-1DST,M3.5.0,M10.5.0)
```

4.5.11 VTAM-APPLID

Use this parameter to define the NCI CM LU 6.2 listener.
Note: NCI CM currently supports only one LU 6.2 listener.

Syntax:

`VTAM-APPLID(`***`LUName`***`)`

LUName char(1-8)
It specifies the VTAM Application ID (LU-Name) of the server. The VTAM Application ID Name must be defined to VTAM as a Primary Logical Unit (PLU) whereby PLU-Name must be identical with ACB-Name.

For each NCI Communication Manager instance that uses a LU 6.2 listener, a VTAM LU-Name has to be be defined. If the NCI Communication Manager hosts application programs that require to run in a separate MVS address space (TP-INSTANCE(PROCESS)), a separate LU-Name must be defined for each TP-Instance.
The naming convention for minor LU-Names is: **6 digits major application name + 2 digits hexanumeric suffix** (e.g. APX1APP1, APX1AP00-APX1AP01,APX1APnn). The maximum number of minor LU-Names is limited to 254 (suffix: 00-FE).

```
APX1APP1  VBUILD TYPE=APPL
*
*        APPL DEFINITION FOR THE APPLICATION APP1
*
APX1APP1  APPL  ACBNAME=APX1APP1,
                APPC=YES,
                DLOGMOD=#INTER,
                PARSESS=YES,
                SECACPT=ALREADYV,
                VPACING=1
APX1AP00 APPL   ACBNAME=APX1AP00,
                APPC=YES,
                DLOGMOD=#INTER,
                PARSESS=YES,
                SECACPT=ALREADYV,
                VPACING=1
APX1AP01 APPL   ACBNAME=APX1AP01,
                APPC=YES,
                DLOGMOD=#INTER,
                PARSESS=YES,
                SECACPT=ALREADYV,
                VPACING=1
APX1APnn APPL   ACBNAME=APX1APnn,
                APPC=YES,
                DLOGMOD=#INTER,
                MODETAB=MODEAPPL,
                ……..
```

Figure 4-7: VTAM definition for NCI LU 6.2 listener

Note: The log mode #INTER is an IBM default LU 6.2 log mode that is shipped with VTAM and should be available in all data centers.

4.5.12 TRACE

Syntax:

TRACE(***NONE/ALL/FUNC/DATA/OPTIONS/PROT/CB/CODESET***)

Enable tracing. The trace messages will be written to DD:NCITRACE.
This parameter should be only used, when requested by the NCI support.

4.5.13 TRUSTED-HOST

Syntax:

TRUSTED-HOST(***ipAddress***)

Definition of hosts that should be treated as trusted based on the client's IP address. Users from those hosts do not have to provide a password for user verification. If a client requests a TP that has been defined with attribute: SECURITY-LEVEL(CHECK/PROPAGATE), the client normally must provide a valid userid and password. If the client's host has been defined as a TRUSTED-HOST, than the client must only provide a valid userid without a password (alreadyverified will be assumed).
The local host itself is implicitly trusted.

ipAddress char(1-60)
The client host's IP address that should be treated as trusted.

4.5.14 tpx-PROCNAME

Syntax:

tpx-PROCNAME(***procedureName***)

Define a specific start procedure for a TP that is defined as “TP-INSTANCE(PROCESS)”.
If the NCI Communication Manager creates a new TP-Instance for such a TP, a new z/OS address space will be created using the given procedure name. If no specific start procedure is defined the start procedure of the NCI CM itself will be used to start new TP instances.
Note: This parameter has no effect on TPs defined with “TP-INSTANCE(TASK)”.

tpx char(1-60)
Name of the TP the given start procedure should be used for.

procedureName char(1-8)
Name of start procedure to use for this TP.

4.6 NCI CM Configuration Module

The NCI configuration module contains application specific information such as TP-Names and TP-properties. Usually the configuration module will be provided by the application developer and is shipped with the application. Normally there is no need to change this module at the data center site. The configuration module is a z/OS load module and is loaded by the server at startup. The configuration module is addressed via the parameter MODULE-NAME(...) in the configuration file. The configuration module has to be created from a source file using the ISPF-Dialog **NCI.**

4.6.1 Creating a NCI Configuration Module using the NCI ISPF dialog

This is a step-by-step description how to assemble a NCI configuration module source file into a configuration module using the NCI ISPF online dialog.

Type NCI on the ISPF command line to invoke the NCI ISPF dialog.

Note: To do this the NCI ISPF libraries SNCIEXEC, SNCIPNL, SNCIMSG and SNCISKL have to be in your ISPF library concatenation or have to be provided via a cumulative library.

```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------
                     ISPF Command Shell

Enter TSO or Workstation commands below:

===> nci


Place cursor on choice and press enter to Retrieve command

=>
=>
```

Figure 4-8: Invoking the NCI ISPF dialog

On the main window, specify the data set names and member name to use.
Note:
(1) The load library must be available (in the search order) of the NCI CM instance that wants to use this configuration module.
(2) Choose a name that is meaningful for your application. If the target load library is a library that is shared with other NCI CM instances, make sure that there are no naming conflicts with other NCI CM instances.

```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------------------------
                              ISPF Command Shell
Ent +-------------------------------------------------------------------+
    |  Actions    Type    Help                                          |
=== |--------------  Edit / Build NCI Sideinformation    -------------  |
    | Command ===>                                                      |
    |                                                                   |
    |                                                                   |
Pla |  Source Library : 'MYAPP.CONFIG'                                  |
    |                                                                   |
=>  |   Input  Member  : MYCMCFG     Blank to display member-list       |
=>  |                                If member does not exist, sample   |
=>  |                                definitions will be provided       |
=>  |                                                                   |
=>  |   Object Library :                                                |
=>  |   (optional)                                                      |
=>  |                                                                   |
=>  |   Load   Library : 'MYAPP.LOADLIB'                                |
=>  |                                                                   |
=>  |                                                                   |
    +-------------------------------------------------------------------+
```

Figure 4-9: NCI ISPF dialog main window

Press ENTER to open the edit window. Adapt or change the the NCI configuration module source according to your application's needs. Use PF3 to leave the edit window and save the changes.

```
EDIT       MYAPP.CONFIG(MYCMCFG) - 01.00                  Columns 00001 00072
 Command ===>                                               Scroll ===> CSR
 ******************************** Top of Data ********************************
000001
000002    ********************************************************************
000003    * Source NCI Configuration Module for APPLICATION: SAMPLE       *
000004    ********************************************************************
000005      AUTO-TERMINATION(NO)
000006    *
000007    *
000008        TP-NAME(TSTTP1)
000009                DEFAULT-TP(NO)
000010                PROGRAM-NAME(NCITSTCS)
000011                PIP-DATA(PARM1,PARM2)
000012                TP-MIN(1)
000013                TP-MAX(10)
000014                CYCLE-TIME(0)
000015                IDLE-TIME(10)
000016                SECURITY-LEVEL(NONE)
000017         LOCAL-CP($IBM273)
000018         TP-INSTANCE(TASK)
000019    *
000020        TP-NAME(TSTTP2)
000021                PROGRAM-NAME(NCITSTCS)
000022                TP-MIN(0)
000023                TP-MAX(1)
000024         SRV-TIMEOUT(10)
000025    *
```

Figure 4-10: NCI ISPF dialog edit window

Select Type->Edit/Build NCI Configuration, to tell the NCI dialog that a NCI configuration module should be build.

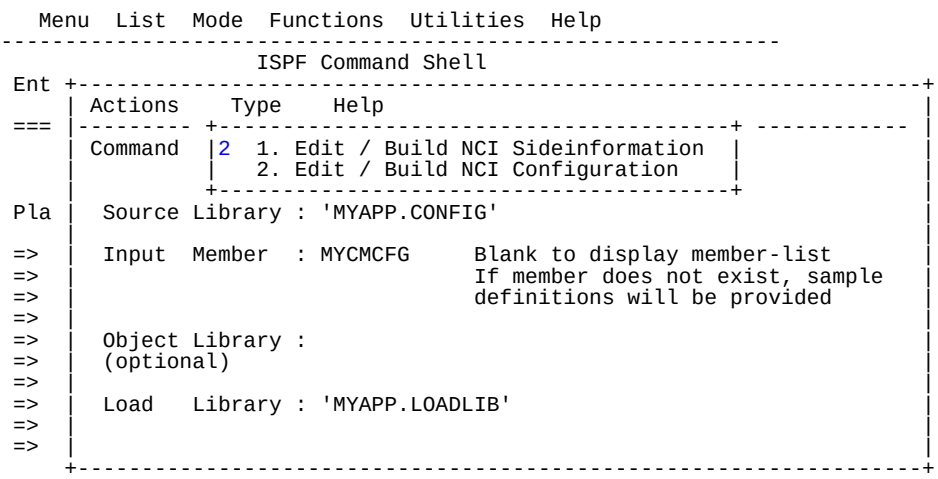

Figure 4-11: NCI ISPF dialog - select type of module to be build

Use the Action BUILD to create the NCI configuration module. The module is placed in the specified load library.

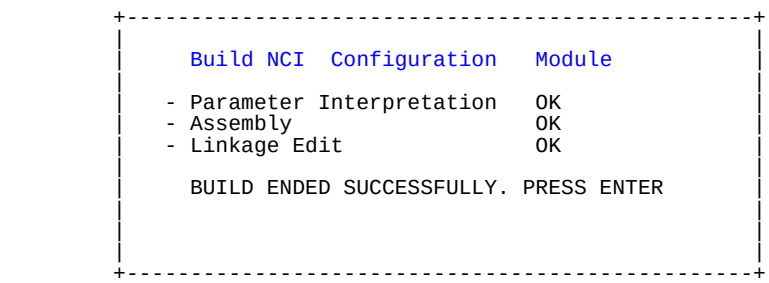

Figure 4-12: NCI ISPF dialog - process status window

The configuration module can now be used.
Whenever you want to change the NCI configuration module you have to redo this complete process.

4.6.2 NCI Configuration Module – Parameter Reference

4.6.2.1 AUTO-TERMINATION

Enable or disable the NCI Communication Manager auto termination feature.

Syntax:

AUTO-TERMINATION(***YES/NO***)

YES/NO
YES: Server ends after the last TP ended.
NO: Server runs until stopped by operator.TERM signal received.

Default value: NO

4.6.2.2 TP-NAME

Syntax:

TP-NAME(***tpName***)

tpName (char 1-60)
Application-specific name that identifies a NCI server program and its properties. A client addresses a specific TP using the NCI API call nciSetServiceId(...).

Default value: n/a

4.6.2.3 DEFAULT-TP

Syntax:

DEFAULT-TP(***YES/NO***)

YES/NO
If set to YES, this TP is the default TP.
If a Client connects to a server without specifying a TP-Name (nciSetServiceId), the TP defined as default will be used. If no TP-Name is defined as default, the client request will be rejected by the NCI Communication Manager.

Default value: NO

4.6.2.4 PROGRAM-NAME

Syntax:

PROGRAM-NAME(***programName***)

programName char (1-8)
Name of application program that is started, if the client requests the TP. The Application program contains application server logic. It have to be written according to the NCI server program specifications.

Default value: n/a

4.6.2.5 PIP-DATA

Syntax:

`PIP-DATA(`***`parameterString`***`)`

`parameterString` char (1-100)
Program Initial Parameters that the application program will receive, if it is started for the first time. The parameters are are passed to the application program as it would have been invoked by the EXEC PGM= batch interface.
For programs written in C this means, the program receives the parameters as argc/argv values.

Default value: n/a

4.6.2.6 TP-MAX

Syntax:

`TP-MAX(`***`number`***`)`

number numeric (1-999)
Number that specifies the maximum number of TPs that should be instantiated for this type of TP. If the maximum number of started TPs is reached, all subsequent client requests for this TP will be queued until a TP gets available.
Note: A TP gets available (free) whenever it has completely finished serving a client.

Default value: 1

4.6.2.7 TP-MIN

Syntax:

`TP-MIN(`***`number`***`)`

number numeric (1-99)
Number that specifies the number of TPs that should be started during NCI Communication Manager startup. Use this feature to provide some TP instances immediately after server startup. This speeds up the first client requests, because the clients do not have to wait until the application server programs have initialized.
Note: TPs started during startup are not affected by idle time expiration. That means they remain active until server shutdown.

Default value: 0

4.6.2.8 CYCLE-TIME

Syntax:

`CYCLE-TIME(`***`timeValue`***`)`

timeValue numeric (1-1440)
Time value in minutes. The cycle time is a NCI internal timer that elapses periodically. Each time this time elapses NciGet() returns control to the application program with the reason code NCI_RCC_CYCLE (12). The application program can then do some periodic work, and should then reinvoke the NciGet() API call.
A value of 0 means that the cycle time is disabled.
Note: Do not use this timer for “real time processing”. The internal time base used by NCI is not as accurate as you may expect.

Default value: 0

4.6.2.9 IDLE-TIME

Syntax:

`IDLE-TIME(`***`timeValue`***`)`

`timeValue` numeric (1-1440)
Time value in minutes. The idle time is a NCI internal timer that elapses whenever the TP instance was idle for the specified period of time.
After a TP instance has finished work it gets IDLE. At this point the idle timer for this instance is activated. If this TP instance will not serve a new client within the idle time period, the idle timer for this TP instance expires. The NciGet() API calls returns control to the application program with reason code NCI_RCC_IDLE (8).
The application program should cleanup and terminate in this case.

Default value: 10

4.6.2.10 SECURITY-LEVEL

Syntax:

SECURITY-LEVEL(***NONE/CHECK/PROPAGATE***)

NONE/CHECK/PROPAGATE

- NONE
  NCI Communication Manager does not perform any security checks. The client does not have to provide any security information. If the client provides security information, it will not be checked.

- CHECK
  Before the client request is passed to the TP, the security information provided by the client will be checked against the z/OS user registry (RACF). The client has to provide a valid userid/password combination or the client host must be defined as trusted, otherwise the request will be rejected.
  Trusted Hosts: The server accepts requests without a password from those clients (hosts) that are defined as already_verified.
  If protocol LU 6.2 is used: attribute already_verified can be restricted on session level via RACF-class: APPCLU.
  If protocol TCP/IP is used: attribute already_verified is restricted to those hosts that are defined with a TRUSTED-HOST directive in the NCI parameter file. See chapter “4.5.13

TRUSTED-HOST".

- PROPAGATE
  Implies CHECK, with the extension that the client userid is propagted to the TP. That means the TP runs under the authority of the client user. Further security checks, for example if the application program accesses data sets are then automatically done against the client userid.

Note: For TPs that will be accessed via the MQ Protocol only SECURITY-LEVEL(NONE) is currently supported.

Default value: NONE

4.6.2.11 TP-INSTANCE

Syntax:

`TP-INSTANCE(`***TASK/PROCESS***`)`

TASK/PROCESS

- TASK
  The TP instance (application program) will be started as a z/OS subtask (thread). This means, multiple application programs may run in parallel within the same NCI server region (z/OS address space). They all share the same server region storage.

- PROCESS
  The TP instance (application program) will be started as a separate z/OS address space. Running a TP as a PROCESS instance produces more overhead compared to a TASK level instance. However it provides the optimal separation between the application programs.

Note: In the past we made the experience, that the most of the existing Cobol programs were not able to run as z/OS subtasks in parallel. Therefore, if you run into trouble with TASK level instances, try to run your application with the PROCESS level.

Default value: TASK

4.6.2.12 LOCAL-CP

Defines the code page for user data conversion on the server side. This is the code page in which the application program receives the data from the NciGet() API call.

Note: The local code page can also be set using the NCI API (NciSetLocalCodePage) or via the LOCALCP setting in the NCI side info file when the application uses the addressing type SIDE.
The order of precedence is as follows:
NCI API (NciSetLocalCodePage)
LOCAL-CP setting in TP definition
LOCALCP definition in NCI side info file
NCI Default

Syntax:

`LOCAL-CP(`***codePage***`)`

codePage

One of the NCI predefined code pages that start with '$' .
These are: `$IBM273, $IBM1047, $ISO88591, $ROMAN8, $CP1252.`
Those code pages are supported on all platforms that are supported by NCI.
The following code page settings have a special meaning:
$STANDARD
Backward compatibility with previous NCI releases.
$AUTO
Server code page is determined automatically using the current LOCALE.

Additionally any user defined code page can be specified. User defined code pages are those code pages that are supported by the z/OS iconv() service. NCI uses the iconv() C/C++ for the data conversion. To get a list of supported code pages please refer to "*z/OS C/C++ Programming Guide*".

Default value: $IBM273

Restriction: This setting has only an effect when using the protocol TCP/IP.

4.6.2.13 NET-CP

Code page of the transmitted data. Usually there is no need to use this setting, because newer NCI clients always send the data in the ISO88591 code page. Therefore the default setting should work properly.

Note: The net code page can also be set using the NCI API (NciSetNetCodePage) or via the NETCP setting in the NCI side info file when the application uses the addressing type SIDE **.**
The order of precedence is as follows:
NCI API (NciSetNetCodePage)
NET-CP setting in TP definition
NETCP definition in NCI side info file
Default

Syntax:

`NET-CP(`***codePage***`)`

codePage
see description of “4.6.2.12

LOCAL-CP”

Default value: $ISO88591

Restriction: This setting has only an effect when using the protocol TCP/IP.

4.6.2.14 MAX-QUEUEDEPTH

Limit the number of client requests that will be queued for this TP.
Client requests will be always queued, if all TP instances of the TP are in use and new client requests enter the server.

Syntax:

MAX-QUEUEDEPTH(***number***)

number numeric (0-n)
Maximum number of client requests that will be queued for this TP. If this limit is reached new client requests will be rejected with the reason code NCI_RCC_TP_ALLINUSE (148). On server side see MSG: NCI254BE.
Note: A value of 0 means, that no client requests will be queued for this TP.

Default value: 45

4.6.2.15 SRV-TIMEOUT

Sets a timeout value on the NCI server side.
Defines the max. time a NciGet() call waits for receiving data from the client.
This timeout only works when NciGet() really does a data receive, and not when NciGet() waits for a new client, because the previous client was completely processed.

Note: The Timeout can also be set from the server application code using the NCI API call NciSetTimeout or the TIMEOUT setting in the NCI side info file when the server application uses the addressing type SIDE. The order of precedence is the following:
NCI API call NciSetTimeout
SRV-TIMEOUT setting in TP definition
TIMEOUT definition in NCI side info file

Syntax:

`SRV-TIMEOUT(`***`timeValue`***`)`

`timeValue` numeric (0-n)
Timeout value in seconds. Defines the max. time a NciGet() waits for receiving data from the client. If the timeout is reached, NciGet() returns with reason code NCI_RCC_TIMEOUT (160).

Default value: n/a, no timeout will be used

Restriction: This setting has only an effect when using the protocol TCP/IP or SSL.

5 NCI Side Info

NCI sideinformation offers the possibility to define application specific NCI API parameters outside the application. We recommend to use the NCI sideinformation, because it provides a standardized way to set NCI parameters for an application. Especially in production environments this is very helpful, to support the standardization process.

If an application wants to use NCI sideinformation, the following NCI API calls must be used to address the sideinfo file/module.

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

For more information about the NCI sideinfo usage please refer to the *"NCI Application Programming Reference"* documentation. Throughout the rest of this chapter only the z/OS specific parts of the sideinfo usage will be discussed.

NCI for Unix and Windows uses a file based sideinformation, however on z/OS the sideinformation has to be available as an MVS load module. To create a sideinformation module, the sideinfo source has to be assembled into a load module. This assembly process is available as an online ISPF dialog or can be called in a batch environment.

5.1 Creating a Sideinfo Module using the NCI ISPF Dialog

This is a step-by-step description how to assemble a NCI sideinfo source file into a sideinfo module using the NCI ISPF online dialog.

Type NCI on the ISPF command line to invoke the NCI ISPF dialog.

Note: To do this the NCI ISPF libraries SNCIEXEC, SNCIPNL, SNCIMSG and SNCISKL have to be in your ISPF library concatenation or have to be provided via a cumulative library.

```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------
                     ISPF Command Shell

Enter TSO or Workstation commands below:

===> nci


Place cursor on choice and press enter to Retrieve command

=>
=>
```

Figure 5-1: Invoking the NCI ISPF dialog

On the main window, specify the data set names and member name to use.
Note:
(1) The load library must be available (in the search order) of the NCI process that wants to use this sideinfo module.
(2) Choose a name that is meaningful for your application. If the target load library is shared with other NCI applications, make sure that there are no naming conflicts with these other applications.

```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------------------------
                            ISPF Command Shell
Ent +-----------------------------------------------------------------+
    |  Actions    Type    Help                                        |
=== |--------------  Edit / Build NCI Sideinformation    -------------  |
    | Command ===>                                                    |
    |                                                                 |
    |                                                                 |
Pla |  Source Library : 'MYAPP.CONFIG'                                |
    |                                                                 |
=>  |   Input  Member  : MYSIDE      Blank to display member-list      |
=>  |                                If member does not exist, sample |
=>  |                                definitions will be provided     |
=>  |                                                                 |
=>  |   Object Library :                                              |
=>  |   (optional)                                                    |
=>  |                                                                 |
=>  |   Load   Library : 'MYAPP.LOADLIB'                              |
=>  |                                                                 |
=>  |                                                                 |
    +-----------------------------------------------------------------+
```

Figure 5-2: NCI ISPF dialog main window

Press ENTER to open the edit window. Adapt or change the the NCI sideinfo source according to your application's needs. Use PF3 to leave the edit window and save the changes.

```
EDIT       MYAPP.CONFIG(MYSIDE) - 01.00                  Columns 00001 00072
 Command ===>                                              Scroll ===> CSR
 ******************************** Top of Data ******************************
000001
000002    ***************************************************************
000003    * Source NCI sideinfo module for application MYAPP              *
000004    ***************************************************************
000006    *
000008        SymbolicName(Connection1)
000009          AddrType(MQ)                      *Use MQ series as protocol
000010          PrimAddrInfo(MQTST)               *Name of Queue Manager
000011          SecAddrInfo(QUEUE.MYAPP.IN)       *Name of Queue
000012          Timeout(10)                       *Timeout = 10sec
000013
000014        SymbolicName(Connection2)
000015          AddrType(TCPIP)                   *Use TCP/IP as protocol
000016          PrimAddrInfo(MYAPP.MVS.TELEKOM.DE)  *Hostname of target NCI CM
000017          SecAddrInfo(8598)                 *TCP/IP port number of target NCI
CM
000018          DataConv(Y)                       *Enable data conversion
000019          Timeout(5)                        *Timeout = 5sec
```

Figure 5-3: NCI ISPF dialog edit window

Select Type->Edit/Build NCI Sideinformation, to tell the NCI dialog that a NCI sideinformation module should be created.

```
   Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------
                  ISPF Command Shell
Ent +-------------------------------------------------------------------+
    | Actions    Type    Help                                           |
=== |--------- +-------------------------------------+ ------------ |
    | Command  |1  1. Edit / Build NCI Sideinformation  |              |
    |          |   2. Edit / Build NCI Configuration    |              |
    |          +-------------------------------------+                 |
Pla |  Source Library : 'MYAPP.CONFIG'                                  |
    |                                                                   |
=>  |  Input  Member  : MYSIDE      Blank to display member-list        |
=>  |                               If member does not exist, sample    |
=>  |                               definitions will be provided        |
=>  |                                                                   |
=>  |  Object Library :                                                 |
=>  |  (optional)                                                       |
=>  |                                                                   |
=>  |  Load   Library : 'MYAPP.LOADLIB'                                 |
=>  |                                                                   |
=>  |                                                                   |
    +-------------------------------------------------------------------+
```

Figure 5-4: NCI ISPF dialog - select type of module to be build

Use the Action BUILD to create the NCI sideinformation module. The assembled module will be placed in the specified load library.

```
+-------------------------------------------------+
|                                                 |
|     Build NCI Sideinformation Module            |
|                                                 |
|   - Parameter Interpretation   OK               |
|   - Assembly                   OK               |
|   - Linkage Edit               OK               |
|                                                 |
|     BUILD ENDED SUCCESSFULLY. PRESS ENTER       |
|                                                 |
|                                                 |
|                                                 |
+-------------------------------------------------+
```

Figure 5-5: NCI ISPF dialog - process status window

The sideinformation module can now be used. Whenever you want to change this sideinformation module you have to redo this complete process.

5.2 Creating a Sideinfo Module using NCI Batch

In a production environment you may prefer using a batch process to automate the assembly of the sideinformation module. The result is the same like using the NCI ISPF online dialog, the sideinfo module is placed in the defined load library. The member NCIBATCH in the NCI samples data set can be used as a sample.

```
//NCIBATCH JOB
//*********************************************************************
//* NCIBATCH                                                          *
//*                                                                   *
//* GENERATE NCI SIDEINFORMATION OR CONFIGURATION MODULE              *
//*                                                                   *
//* PARAMETERS: 1. SIDEINFORMATION | CONFIGURATION                    *
//*             2. SOURCE LIBRARY                                     *
//*             3. OUTPUT LOAD LIBRARY                                *
//*             4. SOURCE MEMBER (SIDEINFORMATION OR CONFIGURATION)   *
//*                                                                   *
//*********************************************************************
//STEP1    EXEC PGM=IKJEFT01,DYNAMNBR=99
//SYSEXEC  DD   DSN=???.SNCIEXEC,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
%NCIBATCH SIDEINFORMATION +
          'MYAPP.CONFIG' +
          'MYAPP.LOADLIB' +
          MYSIDE
/*
```

After the batch job successfully completed the sideinformation module can be used in the application. Whenever you want to change this sideinformation module you have to reassemble it.

6 Setting up NCI for SSL

6.1 What is SSL?

Secure Sockets Layer (SSL) is a communications protocol that provides secure communications over an open unsecure communications network. z/OS System SSL provides SSL support that is used by the NCI SSL implementation.
The SSL protocol was developed by Netscape Communications Corporation. SSL provides data privacy and integrity as well as server and client authentication. Once your NCI CM has an enabled SSL Listener, SSL-enabled NCI clients can communicate securely with the NCI CM using SSL. With SSL, you can easily establish a security-enabled NCI communication over the Internet or on your private intranet. An NCI client that does not support SSL will not be able to connect to the NCI CM.
SSL uses a security handshake to initiate a secure connection between the client and the server. During the handshake, the client and server agree on the security keys they will use for the session and the algorithms they will use for encryption. The client authenticates the server; optionally, the server can request the client's certificate. After the handshake, SSL is used to encrypt and decrypt all of the information passed between client and server and server and client. Also the security info that is part of the NCI protocol will be encrypted.
The number of cipher specs that are supported by z/OS System SSL depend on the version of z/OS System SSL. Newer versions support more and stronger cipher algorithm than older versions. Therefore, before you use a cipher in the NCI configuration, first check, if it is supported by the used version of z/OS System SSL.

To use the NCI SSL communication, some additional setup is necessary. Digital certificates and a keystores that hold the certificates are required. The next chapters briefly describe how to create keystores and certificates for NCI servers and for NCI clients.

Note:
(1) The sample jobs that are provided with NCI describe the process of creating so called "self signed certificates". "self signed certificates" imply a lot of disadvantages concerning revocation and validity, they should only be used for testing purposes. In a production environment we strongly recommend to order certificates from an official certificate authority.
(2) In the NCI documentation we only describe the creation and usage of RACF keystores. However, keystores can be also setup as HFS key data bases. For more information about this, please refer to the z/OS System SSL documentation.
(2) For detailed information about z/OS System SSL, please refer to the z/OS System SSL documentation.

6.2 Setting up NCI clients for SSL

The sample members NCISSLC1 - NCISSLCn in the NCI sample data set SNCISAMP are step by step samples that can be used to create the necessary RACF certificates, keystores (keyring) and configuration directives to setup an NCI client on z/OS for SSL.
The certificates created, are signed by an internal CA (self signed!). You can use them for testing purposes, but you should not use them in a production environment.

The table below shows an overview about the sample jobs. Adapt and run these jobs in the order they are listed. Take a look into the JCL itself, every job contains comments about it's function.
Note: The jobs NCISSLC3 and NCISSLC4 are only necessary, if the client needs a client certificate, because the NCI server has enabled client authentication
(SSL-PEERCERTCHECK,REQUIRED)

Job	Description
NCISSLC1	Create a self signed CA certificate
NCISSLC2	Create the client keyring and connect CA certificate to the keyring.
NCISSLC3	Create and sign the client certificate.

NCISSLC4	Connect client certificate to client keyring
NCISSLC5	Permit the NCI client to access the keyring
NCISSLC7	Update your NCI client configuration to use SSL. This can be done by: • If the client uses NCI sideinformation, changing the client's NCI sideinformation • Chaning the client application code itself.

Figure 6-1: Job overview table to setup clients for SSL.

6.3 Setting up NCI CM for SSL

The sample members NCISSLS1 - NCISSLSn in the NCI sample data set SNCISAMP are step by step samples that can be used to create the necessary RACF certificates, keystores (keyring) and configuration directives to setup an NCI CM server for SSL.
The certificates created, are signed by an internal CA (self signed!). You can use them for testing purposes, but you should not use them in a production environment.

The table below shows an overview about the sample jobs. Adapt and run these jobs in the order they are listed. Take a look into the JCL itself, every job contains comments about it's function.

Job	Description
NCISSLS1	Create a self signed CA certificate Note: You do not have to run this job, if you already created the CA certificate in the previous chapter.
NCISSLS2	Create the server keyring
NCISSLS3	Create and sign the server certificate
NCISSLS4	Connect server certificate to server keyring
NCISSLS5	Permit the NCI CM server to access the keyring
NCISSLS6	Update the NCI CM config file to define an NCI SSL listener. Refer to chapter "4.5.10-SSL-PORT" to get detailed information about the listener parameters.
NCISSLS7	Verify that you can establish a secure connection

Figure 6-2: Job overview table to setup a NCI CM for SSL.

7 Installation/Customization Verification

7.1 Verification Test (MQSeries)

Note: A MQSeries infrastructure is required to do this test. The used queue name has to be defined.

Start NCI echo server using sample job NCIPONG.

```
//NCIPONG JOB
//NCIPONG EXEC PGM=NCIPONG,
//    PARM='-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 successfully the following messages appear in the job log.

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

Start NCI echo client by sample JOB (NCIPING).

```
//NCIPING JOB
//NCIPING EXEC PGM=NCIPING,
//    PARM='-a MQ -2 queuename -y Y'
```

Option	Description
-a MQ	AddressingType: MQSeries. Use MQSeries as protocol.
-2 queuename	MQSeries queue name of the NCI echo server (eg. SYSTEM.DEFAULT...)
-y Y	Message requires a reply. That means that the echo server has to send a reply message.

If the NCI echo client completes successfully the following messages appear in the job log.

```
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
```

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

7.2 Verification Test (TCP/IP) Parallel Server

Start NCI Communication Manager using sample job NCICM.

```
//NCICM     JOB
//NCICM     EXEC PGM=NCICM
//NCIPARAM  DD   *
   MODULE-NAME(NCICONF)
   IP-PORT(3450)
```

Option	Description
MODULE-NAME	Name of the NCI configuration module (NCICONF). NCI is shipped with this sample configuration module that is available as source and in translated format.
IP-PORT	TCP/IP portnumber (56789). Port number that is used by the NCI CM TCP/IP listener to accept incoming TCP/IP requests. **Note:** The Make sure, that the port number you choose is not already used by another application. Port 3450 is only used as a sample in this case. You could also choose any other number. Contact your TCP/IP administrator if you aren't sure what port to use.

If the NCI Communication Manager starts successfully the following messages will be written to the system console, respectively the job log.

```
NCI2706I NCICMIP  LST3450: TCP/IP communication INADDR_ANY:3450 established
```

Start NCI echo client using sample job NCIPING.

```
//NCIPING JOB
//NCIPING EXEC PGM=NCIPING,
//    PARM='-a TCPIP -1127.0.0.1 -2 3450'
```

Option	Description
-a TCPIP	AddressingType: TCP/IP
-1 127.0.0.1	TCP/IP address of NCI server. In this case we use the loopback address of the own host.
-2 portnumber	TCP/IP portnumber of the NCI server. The port number has to match the number that was defined in the configuration file of the NCI server.

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 (40 bytes): Response ...: Hello Server
nciping - nciClose ReturnCode: 0
```

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

Stop the NCI Communication Manager using the operator STOP command. While NCI Communication Manager stops the following messages will be written to the system console.and job log.

```
NCI2550I NCICM    : STOP REQUESTED.
NCI2549I NCICM    : STOP REQUESTED. GOING TO STOP ALL ACTIVE TPS.
NCI2650I NCICMIP  LST3450: TCP/IP listener is going to terminate. Reason: got order
to stop.
NCI2512I NCICM    : STOPPED BY OPERATOR
```

7.3 Verification Test LU 6.2 Parallel Server

Start NCI Communication Manager using the sample job NCICM.

```
//NCICM     JOB
//NCICM     EXEC PGM=NCICM
//NCIPARAM  DD   *
   MODULE-NAME(NCICONF)
   VTAM-APPLID(APX1STST)
```

Option	Description
MODULE-NAME	Name of the NCI configuration module (NCICONF). NCI is shipped with this sample configuration module that is available as source and in translated format.
VTAM-APPLID	VTAM Application Id (LU-Name). LU name that is used by the NCI CM LU 6.2 listener to accept incoming LU 6.2 requests. Note: Make sure that the VTAM server LU definitions as well as the VTAM client definitions for NCI clients have been setup in your location. The LU name APX1STST is only a sample name, you may choose any other name.

If the NCI Communication Manager starts successfully the following messages will be written to the system console and job log.

```
NCI2575I NCICM    : LU 6.2 COMMUNICATION (LU:APX1STST) ESTABLISHED
```

Start NCI echo client using sample job NCIPING.

```
//NCIPING JOB
//NCIPING EXEC PGM=NCIPING,
//    PARM='-a LU62 -2 APX1STST'
```

Option	Description
-a LU62	AddressingType: LU 6.2
-2 LU-Name	VTAM Application Id (LU-Name) of the NCI Communication Manager(e.g. APX1STST)

If the NCI echo client completes successfully the following messages will be displayed in the job log.

```
nciping - nciOpen ReturnCode: 0
nciping - nciSetAddrType ReturnCode: 0
nciping - nciSetSecAddrInfo ReturnCode: 0
nciping - sending message: Hello World
nciping - received reply message (40 bytes): Response ...: Hello Server
nciping - nciClose ReturnCode: 0
```

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

Stop the NCI Communication Manager using the operator STOP command. While NCI Communication Manager stops the following messages will be written to the system console.and job log.

```
NCI2550I NCICM    : STOP REQUESTED.
NCI2549I NCICM    : STOP REQUESTED. GOING TO STOP ALL ACTIVE TPS.
NCI2512I NCICM    : STOPPED BY OPERATOR
```

7.4 Verification Test MQSeries Parallel Server

Start NCI Communication Manager using the sample job NCICM.

```
//NCICM     JOB
//NCICM     EXEC PGM=NCICM
//NCIPARAM  DD   *
   MODULE-NAME(NCICONF)
   MQ-APPL-QUEUE(SAMPLE.APPL.QUEUE,MQXX)
```

Option	Description
MODULE-NAME	Name of the NCI configuration module (NCICONF). NCI is shipped with this sample configuration module that is available as source and in translated format.
MQ-APPL-QUEUE	MQSeries Application Queue and Queue Manager. MQSeries queue that is used by the NCI CM MQSeries listener to accept incoming MQseries requests. **Note:** Make sure that the specified MQSeries queue and queue manager have been setup in your location. The queue name SAMPLE.APPL.QUEUE as well as the queue manager name MQXX are only sample names, you may choose any other name.

If the NCI Communication Manager starts successfully the following messages will be written to the system console and job log.

```
NCI5064I NCICMM    : MQSeries MSG-driven TP Listener(Qmgr:…, ApplQ: …) Started
```

Start NCI echo client using sample job NCIPING.

```
//NCIPING JOB
//NCIPING EXEC PGM=NCIPING,
//    PARM='-a MQ -1 MQXX -2 SAMPLE.APPL.QUEUE'
```

Option	Description
-a MQ	AddressingType: MQSeries
-1 Qmgr-Name	Name of the queue manager that owns the application queue of the NCI Communication Manager(e.g. MQXX)
-2 Queue-Name	Name of the application queue used by the listener for message-triggered TPs (e.g. SAMPLE.APPL.QUEUE)

If the NCI echo client completes successfully the following messages will be displayed in the job log.

```
nciping - nciOpen ReturnCode: 0
nciping - nciSetAddrType ReturnCode: 0
nciping - nciSetSecAddrInfo ReturnCode: 0
nciping - sending message: Hello World
nciping - received reply message (40 bytes): Response ...: Hello Server
nciping - nciClose ReturnCode: 0
```

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

Stop the NCI Communication Manager using the operator STOP command. While NCI Communication Manager stops the following messages will be written to the system console.and job log.

```
NCI2550I NCICM    : STOP REQUESTED.
NCI2549I NCICM    : STOP REQUESTED. GOING TO STOP ALL ACTIVE TPS.
NCI2512I NCICM    : STOPPED BY OPERATOR
```

Index

VTAM-APPLID 30

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

T Systems