Natural

TP Monitor Interfaces

Version 9.2.2

December 2024

ADABAS & NATURAL

This document applies to Natural Version 9.2.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: NATMF-TPMON-922-20241211

Table of Contents

Preface

Natural provides interfaces that allow the Natural nucleus to access a TP monitor for online transaction processing and an operating system (OS) for batch processing.

For general information, see also the section *TP/OS Interface* in the *Natural System Architecture* documentation.

This documentation provides detailed information on the operation of Natural with the supported TP monitor systems.

Using Natural with TP Monitors	Provides general information on the usage of Natural with TP Monitors.
Natural under CICS	Describes the functionality of Natural CICS Interface and the operation and individual components of Natural in a CICS environment.
Natural under Com-plete/SMARTS	Describes how to operate Natural in a Com-plete/SMARTS environment.
Natural under IMS TM	Describes how to run Natural under IMS TM.
Natural under TSO	Comprises general information about the Natural TSO Interface and data sets.

Notation *vrs* **or** *vr*

When used in this documentation, the notation *vrs* or *vr* represents the relevant product version (see also *Version* in the *Glossary*).

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://containers.softwareag.com/products** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software GmbH products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Using Natural with TP Monitors

2 Using Natural with TP Monitors

TP Monitor Systems Supported by Natural

Natural supports the following teleprocessing monitor systems:

- **CICS**
- **Com-plete**
- **IMS TM**
- **TSO**

For information on using Natural with a specific TP monitor, refer to the TP monitor interface descriptions in this documentation.

Using Natural in a Teleprocessing Environment

This section covers the following topics:

Embedding Natural in a TP Environment

In a teleprocessing monitor environment, Natural operates as a standard TP program and follows the rules that apply to programs executing under the control of this TP monitor.

As the Natural code is fully reentrant, it is shared between all Natural users and only a work area exists on an individual per-user basis (and only for the duration of this user's Natural session).

Natural user programs (transactions) can be executed together with native TP programs to form an integrated system comprising both Natural and conventional programs.

Relevant Natural Profile Parameters

There are various Natural profile parameters that apply if Natural is used with a TP monitor.

For an overview of these parameters, see *TP Monitor Interfaces* in *Profile Parameters Grouped by Category* in the Natural *Parameter Reference* documentation .

Calling Natural Transactions under a TP Monitor

The Natural transactions can be called by invoking the TP program called Natural and supplying the system command `LOGON` and the name of the Natural transaction to be executed in the stack.

Multiple commands/transactions and input data for the commands/transactions can be passed using the stack when calling Natural.

Monitoring and Controlling TP-Monitor-Specific Natural Characteristics

The Natural utility `SYSTP` provides various functions which can be used to monitor and control characteristics of Natural that are specific to TP monitors.

`SYSTP` is available under the TP monitors Com-plete, CICS, IMS TM, and TSO.

For further information, see *SYSTP Utility*.

Terminating a Natural Session

The Natural session can be terminated by executing the Natural statement `TERMINATE` or the system command `FIN`.

Example Programs

The Natural library `SYSEXTP` contains several example programs for specific functions that apply only under certain TP monitors.

II Natural under CICS

This document describes the functionality of Natural CICS Interface (product code NCI) and the operation and individual components of Natural in a CICS environment.

Notation *vrs* or *vr*:

When used in this document, the notation *vrs* or *vr* represents the relevant product version (see also *Version* in the *Glossary*).

Related Topics:

- *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation

- *Natural under CICS Abend Codes and Error Messages*, *Natural under CICS Informational Messages* and *NCISCPRI Warnings and Error Messages* in the Natural *Messages and Codes* documentation
- *SYSTP Utility* - this Natural utility provides various TP monitor-specific functions.
- *Natural as a Server under CICS*

3 Support for zIIP under CICS

Natural CICS Interface supports IBM's System z Integrated Information Processors (zIIPs) in a CICS environment on z/OS.

For information on Natural for zIIP support and required prerequisites, see the *Natural for zIIP* documentation.

For the changes in installation, see *Installing Natural CICS Interface on z/OS* and *Installing Natural for zIIP on z/OS*.

4 Natural CICS Interface Functionality

This chapter describes the functionality of Natural CICS Interface.

Natural CICS Interface

Natural CICS Interface controls session initialization, roll-in restart (in pseudo-conversational mode), terminal I/O, database access, ABEND processing, Natural local buffer pool calls and the loading, linking to and releasing of external subroutines. In addition, all roll I/O operations are performed by Natural CICS Interface.

Natural Environment-Dependent Nucleus for CICS

The Natural environment-dependent nucleus (described in the Natural *Installation* documentation) for CICS consists of the following components:

- The object module `NCINUC`.

  This module holds Natural CICS Interface system control logic and the logic required for calling operating-system and CICS services.

  This module also holds the entry routine, which in particular prepares Natural CICS Interface Language Environment (LE) linkage; see ***Natural CICS Interface and IBM Language Environment (LE)***.

- The `NTCICSP` macro of the Natural parameter module (see *CICSP - Environment Parameters for Natural CICS Interface* the *Parameter Reference* documentation).

  This macro holds Natural CICS Interface parameters required for runtime and system environment generation options. The module is not CICS version dependent, although some of the parameters should be set depending on the CICS version.

- The object module `NCIXCALL`.

  This module is a separate program in CICS, that is, it is not linked to the Natural nucleus, as it is invoked via `EXEC CICS LINK` from 3GL programs called by Natural; see *Natural 3GL CALLNAT Interface* in the *Operations* documentation. The module is independent of the CICS version.

CICS Shutdown under Natural

The Natural environment-dependent nucleus is eligible to be placed into the CICS PLTSD for CICS quiesce stage 1 or 2 execution.

- When executed in quiesce stage 1, Natural CICS Interface force-terminates all active Natural sessions prior to performing the SYSTP snapshot function (described in *SYSTP Utility* in the Natural *Utilities* documentation).
- When executed in quiesce stage 2, Natural CICS Interface performs the SYSTP snapshot function.

Natural CICS Interface holds logic to be called (via a CICS LINK) by a node error program with the relevant CICS terminal entry address either in the CICS COMMAREA.

System Control under CICS

Natural features specific to CICS include the organization of dynamic storage in threads and the additional capability of handling these threads so that the Natural CICS System Control Program can more efficiently handle dynamic storage.

The Natural CICS System Control Program was initially developed to overcome the 64 KB GETMAIN limit under CICS. It provides complete storage allocation and management functions, including roll file I/O operations and relocation functions for pseudo-conversational users.

In order to enhance the pseudo-conversational processing capabilities of Natural with CICS, the System Control Program uses threads, a contiguous amount of storage which is set up for each user. This structure allows Natural to manage dynamic storage with minimal CICS involvement.

A complete understanding of system control can be attained from the following discussion of its structure and operation. Ensure that you understand this mechanism before starting the installation procedure of Natural under CICS.

Natural Components in CICS Dynamic Storage

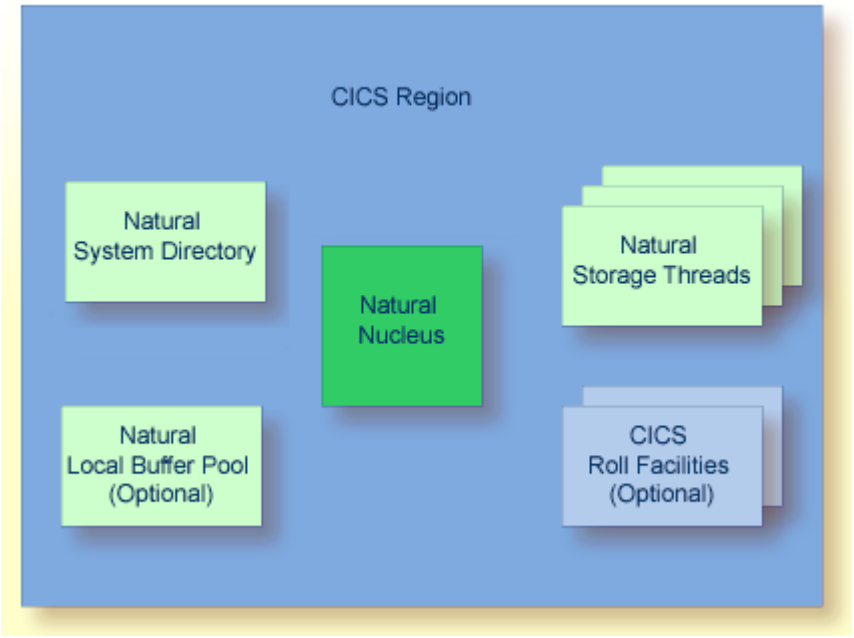
Scenario 1:

Single CICS Region

The diagram below shows the components of the Natural system that reside in CICS dynamic storage. The components are explained under the following headings:

- ***Natural Storage Threads under CICS***
- ***Natural Local Buffer Pool under CICS***

Natural Roll Facilities



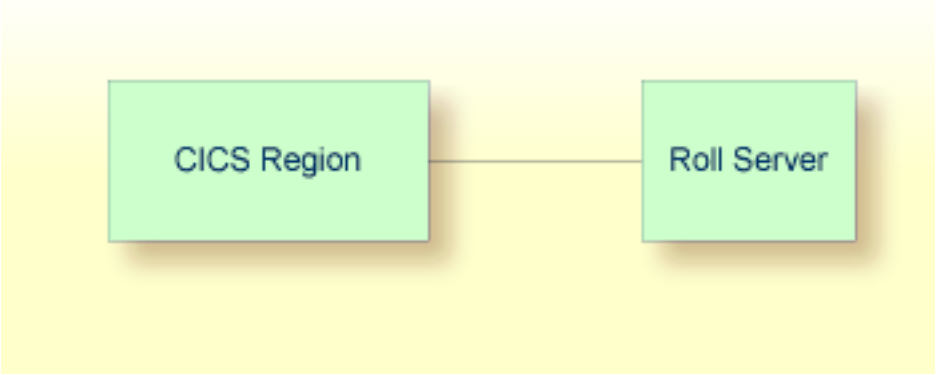
Scenario 1 applies when running Natural locally in a single CICS application region.

Note for z/OS:

Additional scenarios are possible. The following three diagrams show combinations of z/OS systems, CICS regions, the *Natural Roll Server* and the *Natural Authorized Services Manager* (described in the *Operations* documentation).

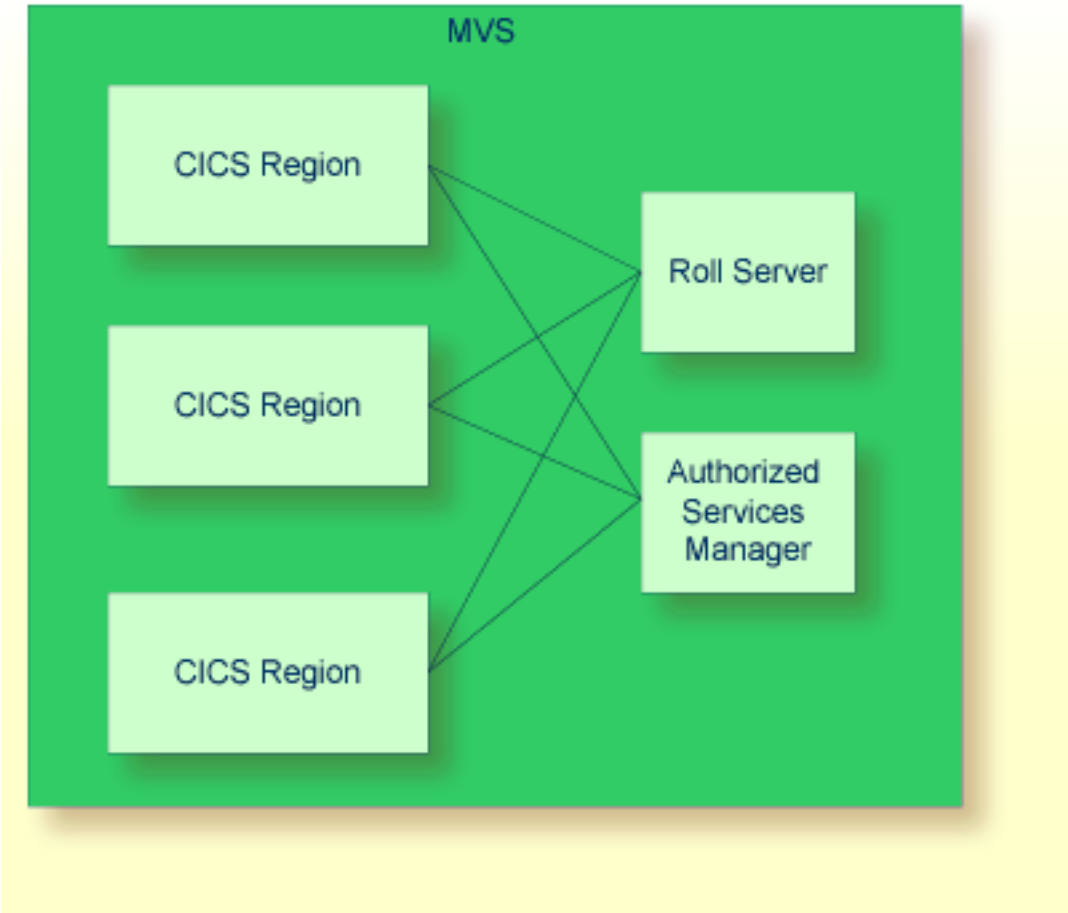
Scenario 2:

Single z/OS With Single CICS Region, Single Roll Server



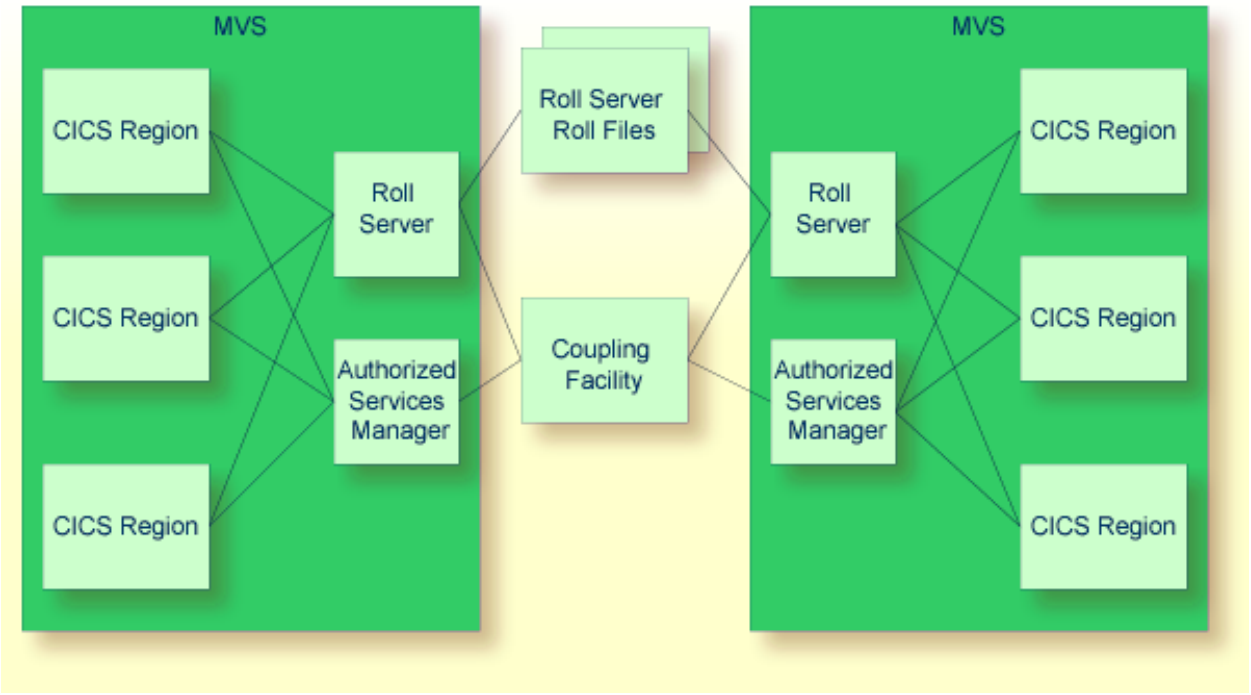
Scenario 3:

Single z/OS With Multiple CICS Regions, Single Roll Server and (Optional) Authorized Services Manager



Scenario 4:

Multiple z/OS With Multiple CICS Regions, Multiple Roll Servers/Authorized Services Managers



Parameter Settings Required for the Above Scenarios

Module	Scenario 1	Scenario 2	Scenario 3	Scenario 4
NTBPI (BPI)	TYPE=NAT,SIZE=*nnn*	n/a	n/a	n/a
NCMDIR CICSPLX	NO	NO	YES/MODE	YES/MODE
NCMDIR SIPSERV	NO	NO/YES	yes	yes
NCMDIR ROLLSRV	NO	yes	yes	yes
Roll Server CF structure name	n/a	none	none	*name*
Authorized Services Manager/SIP	n/a	n/a	SIP slot number/size	XCF group name/CF structure name

Natural CICS Interface requires a SIP slot size of 256 bytes.

Note: For the scenarios 2, 3 and 4, the very first Natural session initializing the NCI environment must have the SUBSID parameter set to the value of the corresponding *Roll Server* and/or *Authorized Services Manager*.

Natural Storage Threads under CICS

A thread is a contiguous storage area from where Natural requests all its required storage. It can either be storage shared by several Natural users or, in 31-bit mode environments, CICS user storage above the 16 MB line dedicated to a specific task.

Each storage thread can be seen as the "address space" for a Natural user. Each memory allocation request issued by the Natural nucleus is transferred to the system control program to be satisfied from the storage thread.

Storage threads are allocated when Natural CICS Interface is initialized. They are allocated in a CICS region or partition, in which case they are permanent (shared) threads or they are allocated during the start of a Natural CICS task, in which case they are exclusive threads (task-dependent user storage).

The technique of storage threads was implemented with Natural for the following reasons:

- To overcome the 64 KB limitation of CICS for user storage in non-31-bit mode systems.
- To be able to optimize rolling (formerly, each piece of user storage had to be written to the roll medium; now, as there is a contiguous storage area, this area is compressed by making the relevant portions contiguous to each other before rolling out).
- Natural CICS Interface tries to satisfy all `GETMAIN` requests of a Natural session from its thread. This is faster than `GETMAIN` requests by means of CICS service calls. This is particularly true for CICS command level calls, as the CICS `EXEC` Interface Program (`EIP`) is involved, too.

A thread is released by the owning task with every screen I/O. This is true for both conversational and pseudo-conversational tasks. When a session is resumed, its storage is rolled into a thread again, unless its storage is still there; that is, no other task used the thread in between.

The Natural thread selection algorithm balances thread usage to minimize roll I/O operations. This means that the more threads there are, the better is the chance of finding the old data thus preventing a roll-in. However, the more threads there are, the more paging the operating system must perform to keep all threads efficiently in real storage.

Threads are grouped together depending on their size and their type; that is, whether they have been pre-allocated as permanently shared storage or via a `GETMAIN` request. The decision on which kind of thread group to use, is controlled by the CICS transaction code at session initialization time. All storage threads belonging to the same group have the same size.

The thread should be defined as small as possible; see also the *Buffer Usage Statistics* function of the SYSTP utility described in the Natural *Utilities* documentation. However, the thread must still be large enough to hold the session with the largest sizes.

If you have separate Natural development and production environments, the rule is to have more smaller threads in the production environment (to serve production requests as soon as possible) and fewer larger threads in the development environment (as Natural programmers normally need larger Natural sizes and have longer “think times”).

The very first Natural session allocates all permanent (shared) threads.

Natural Roll Facilities under CICS

As permanent storage threads are shared by several users and as larger threads allocated via `GETMAIN` should not be kept for too much time, a Natural task releases its thread with each terminal I/O. Previously, however, the user data have to be saved to be able to restart the Natural session after the terminal I/O has been performed.

Session data can be saved by using

- the Natural Roll Server with its local roll buffer and roll files;
- the CICS Roll Facilities.

See also the various **component scenarios**. For more information, see *Roll Server* in the Natural *Operations* documentation.

CICS Roll Facilities

CICS Roll Facilities are local CICS storage facilities. They can be either CICS main or auxiliary temporary storage or VSAM relative record data sets (RRDS) which the user has previously defined to CICS. These files allow Natural to store a user's compressed dynamic storage when a roll-out occurs.

Every CICS service request causes CICS system overhead. So, the larger the `CISIZE`/record size for the roll facility is, the less CPU overhead occurs due to fewer CICS service calls to roll a Natural session. On the other hand, larger `CISIZE`/record size also means more VSAM buffer space allocated for the roll facility.

See ***Performance Considerations*** for further information on roll facilities.

Caution: When using the Roll Server, the CICS Roll Facilities are not available.

Natural Local Buffer Pool under CICS

The Natural local buffer pool contains all Natural modules during execution and copies of Natural modules once they have been loaded from the Adabas or VSAM system file.

The local buffer pool must be large enough to minimize the number of Natural program loads. However, if the local buffer pool is too large, this means wasted storage and may introduce paging overhead.

The local buffer pool is allocated as `GETMAIN` storage, that is, `EXEC CICS GETMAIN SHARED` with all CICS Transaction Server versions. Sufficient storage must be available in the partition or in the relevant CICS DSA.

A local buffer pool is optional, as Natural can also run with a global buffer pool, which can be shared with other Natural environments such as *Natural in Batch Mode*, ***Natural under TSO*** or ***Natural under IMS***.

Natural CICS Interface System Control Records in CICS Temporary Storage

Natural CICS Interface remembers its permanent `GETMAIN`ed storages, that is, storages acquired via `EXEC CICS GETMAIN SHARED` in NCI system control records in CICS main temporary storage.

These system control records are kept for two reasons:

1. System recovery:

   As all NCI related storages are chained of the NCI system directory, the system control records can be used to re-construct storage chains in case of storage corruptions.

2. Clean up old NCI system after CICS `NEWCOPY` of NCI system directory module:

   At NCI system environment initialization, NCI checks for existing system control records, and, if found, NCI frees the associated permanent storages prior to the installation of the new environment.

The CICS temporary storage queue names of these control records are *prefixX*CR, where *prefix* is the common prefix for Natural CICS components (see `NTCICSP` macro parameter `PREFIX`) and *X* is a hexadecimal value, namely

x'01'	for the main system control record holding information about NCI system directory extension, shared threads (TYPE=SHR), and secondary SIR blocks (see NCMDIR generation parameter USERS).
x'02'	for the parms system control record holding information about the NCI shared profile parameters retrieved via file input (see the PRMDEST parameter of the NTCICSP macro).
x'03'	for the pools system control record holding information about all local pools belonging to the NCI environment.

Important: As the NCI system control records describe a local NCI environment, these CICS MAIN temporary storage queues must be kept also in the CICS AOR. This is particularly true when running Natural in CICSPlex.

NCIDIREX - System Directory Module Name Exit Interface

The name of Natural CICS Interface system directory module is *prefix* CB by default (see PREFIX parameter of the NTCICSP macro) unless specified explicitly via the DIRNAME parameter of the NTCICSP macro.

The NCIDIREX exit interface is to set/modify the name of the Natural CICS Interface system directory module at run-time. This makes it possible to use the same NCI driver/ Natural parameter module, but use different NCI environments (thread groups/thread sizes, etc) by accessing different system directory modules, depending for example on CICS system ID, transaction ID.

The first 5 characters of the directory module name are also used as part of CICS temporary storage queue names related to the relevant NCI environment. So when running more than one Natural CICS environment in a CICS region, the relevant system directory module names must be different in the first 5 characters.

The NCIDIREX interface exit is called using standard linkage conventions (Registers 13, 14, 15 and 1) but in addition with Registers 4 and 5 holding CICS EIB and EISTG addresses to enable the exit to call CICS services.

Source module XNCIDIRX contains a sample system directory module name exit.

NCIDTPEX - DTP Terminal I/O Exit Interface

Natural sessions may also be executed using distributed transaction processing (DTP), that is, using APPC or MRO conversations. Formally, such Natural sessions have a terminal associated (CICS TCTTE), however, this is a terminal out of a pool (see CICS SESSIONs / CONNECTIONs) and the “terminal” may change from Natural dialog step to dialog step, that is, such “terminals” cannot be used as key to save a session's context over a “screen I/O”. Because of this nature, such Natural sessions are treated by default as asynchronous sessions (TTYPE=ASYN/ASYL), and Natural does not deal/communicate with these terminals, as they are no 3270 devices.

However, there is an exit interface NCIDTPEX available, which allows you to run the Natural session in a "conversational way":

- when the exit is available, Natural sets up a terminal session (TTYPE=3270);
- Natural terminal input and output operations (RECEIVE/SEND/CONVERSE) are *not* handled by Natural, but passed to the exit for further processing.

The source modules XNCIDTPX and XNCITIOX contain samples of DTP terminal exits.

Control Use of NCIDTPEX

You can set the FDTPX generation parameter of the NTCICSP macro to ON to cause a potential DTP exit to be invoked for all terminal types. This can be helpful, for example, if you want to analyze terminal output before a EXEC CICS SEND operation is executed, or if you want to suppress screen I/O.

NCITIDEX - Terminal ID Exit Interface

The 4-character CICS terminal ID which is unique per CICS region is used by Natural CICS Interface as part of the session key (SIP server, roll server, CICS temporary storage queues). For compatibility with Natural, Natural CICS Interface uses an 8-character field. This NCI terminal ID can be made unique over several CICS regions by appending the CICS system ID to the CICS terminal ID (see UNITID parameter of the NTCICSP macro).

Alternatively, the NCITIDEX terminal ID exit interface can be used to set that NCI terminal ID. It should be noted that for CICS purposes (for example, temporary storage queue names, etc) just the first four characters of the NCI terminal ID are taken. Therefore these 4-character strings must be unique.

The NCITIDEX exit interface is particularly interesting for session managers under CICS in order to distinguish multiple Natural sessions running at the same physical terminal.

The terminal ID set by a NCITIDEX exit is used "externally" by Natural CICS Interface and is the default for the Natural system variable *INIT-ID for Natural use. (The *INIT-ID system variable can subsequently be modified by the NCIUIDEX / NATUEX1 user ID exit interface.)

The NCITIDEX interface exit is called by using standard linkage conventions (Registers 13, 14, 15 and 1), but in addition by using the Registers 4 and 5 holding CICS EIB and EISTG addresses to enable the exit to call CICS services.

Source module XNCITIDX contains a sample terminal ID exit.

Restrictions

Certain Natural CICS Interface functions cannot work if the first four characters of the logical terminal ID do not match the physical terminal.

As a consequence,

- you cannot send a message to a logical terminal by way of message switching,
- you cannot use the SYSTP utility or NEP to flush a session at a logical terminal.

NCIUIDEX - User ID Exit Interface

Natural provides the NATUEX1 user exit interface to determine whether or not a user is authorized to use Natural and to set various Natural system variables.

Whenever a Natural user session is started, the NATUEX1 interface exit is called using standard linkage conventions (Registers 13, 14, 15 and 1).

In a CICS environment, the standard linkage conventions are not sufficient in order to issue CICS service calls and to obtain addressability of CICS control blocks.

Therefore, Natural CICS Interface delivers the load module NCIUEX1 as a NATUEX1 interface exit in a CICS environment. This module just sets up addressability in CICS and calls the NCIUIDEX interface exit by using standard linkage conventions (Registers 13, 14, 15 and 1), but in addition by passing CICS related addresses in other registers: R4 (EIB), R5 (EISTG), R6 (TCTTE).

Thus, if you want to issue requests requiring addressability of the CICS environment, the NCIUIDEX user ID exit interface should be used rather than the standard NATUEX1 interface.

Source module XNCIUIDX contains a sample user ID exit.

Important: With each installation of a new CICS release, the NCIUIDEX interface exit must be reassembled and linked.

NCIXIDEX - Transaction ID Exit Interface

By default, Natural always uses the transaction ID the pseudo-conversational session was started with. This transaction ID can be changed within Natural by using CALLNAT CMTRNSET (library SYSEXTP). The NCIXIDEX transaction ID exit interface can also be used to change the Natural pseudo-conversational transaction ID.

The NCIXIDEX interface exit is called by using standard linkage conventions (Registers 13, 14, 15 and 1), but in addition by using the Registers 4 and 5 holding CICS EIB and EISTG addresses to enable the exit to call CICS services. Source module XNCIXIDX contains a sample transation ID exit.

Note: The transaction ID exit is only invoked prior to pseudo-conversational screen I/O under control of Natural CICS Interface; that is, the exit is not invoked for conversational screen I/O (for example, SET CONTROL 'N') or when Natural is invoked from a front-end program via EXEC CICS LINK.

Natural CICS Interface Debugging Facilities

The following topics are covered:

- Using the TPF Parameter
- Using Asynchronous Natural Sessions

Using the TPF Parameter

The dynamic parameter TPF=(TPF1,TPF2,TPF3,TPF4,TPF5,TPF6,TPF7,TPF8) can be set for driver-specific options by specifying "1" for the corresponding option.

Supported options are:

TPF1	Invoke Adabas linkage module via EXEC CICS LINK with Adabas parameter in TWA and CICS COMMAREA rather than via DCI. Enables debugging of Adabas-related problems via CEDF.
TPF2	With this parameter setting, all CMTRACE trace records are written to the Natural CICS Interface message destination (see the MSGDEST parameter of the NTCICSP macro) in addition to the CICS trace (message number NCI0110). **Note:** Parameter ETRACE=ON or ETRACE=(ON,NOGTF) is required to write trace records.
TPF3	Dump the whole Natural buffer pool. With this parameter setting, the entire Natural buffer pool is included in a CICS transaction dump. **Note:** Usually the Natural buffer pool is not required in a dump, as all objects from the buffer pool relevant to a session are dumped anyway; so this option may only be required in the case of a buffer pool problem.
TPF4	Dump the whole EDITOR buffer pool. With this parameter setting, the EDITOR buffer pool is included in a CICS transaction dump.

TPF6	Handle terminal I/O errors by NCI. With this parameter setting, NCI will not pass control back to Natural for terminal I/O errors, but will handle it by itself, which results in one of the error messages NT06 - NT13.
TPF7	Force abend in case of NCI system errors. With this parameter setting, a program check is forced in case of NS*xx*, NI*xx*, NR*xx* or NUS*nnnn* error messages. This is particularly helpful when a debugging tool intercepting abends is active. Then the error can be analyzed directly online.

When specifying 0 (which can also be omitted), the corresponding option is not set, for example:

TPF=(0,0,0,1) which is equivalent to TPF=(,,,1)

Using Asynchronous Natural Sessions

If the first 5 characters in the dynamic parameter string for starting Natural are ASYN,, Natural CICS Interface will always setup an asynchronous Natural session, regardless of whether a terminal or non-terminal session is started.

This may be helpful for testing purposes, particularly with EDF or with other debugging tools installed.

Natural CICS Interface CICS TWA Usage

The Natural transactions are all defined with a TWA size of 128 bytes, although Natural CICS Interface just uses the first 88 bytes of the CICS transaction work area (TWA) for Natural processing of the following functions:

- on calling Adabas for the Adabas parameter list (up to 32 bytes), Natural CICS Interface saves the TWA contents before calling Adabas and restores it after the Adabas call.
- on calling external programs for the parameter list address pointers (up to 20 bytes, see the Natural CALL statement), Natural CICS Interface saves the TWA contents before calling the external program and restores the TWA call portion after the external program call.
- on invoking a back-end program for the termination message and potential termination data (80 bytes, see *Back-End Program Calling Conventions* in the Natural *Operations* documentation).
- on returning control to a "LINK" front-end caller for the termination message and potential termination data at session end and the termination message area fully reset to low-value at Natural dialog step end respectively, that is, 80 bytes at session and dialog step end.
- for passing LE information at CICS task start (up to 88 bytes, just at start of task).

User programs (front-end, back-end, called external programs) can also take advantage of the CICS TWA to communicate besides Natural, but they should not use the TWA portion used by

Natural; for such cases, it is higly recommended to increase the TWA size of the Natural transactions and use TWA portions outside the first 128 bytes.

5 Natural CICS Generation Parameters

This chapter describes the Natural CICS generation parameters.

For the parameters used to start the Natural CICS, see the section *CICS Startup Parameters* in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.

Related Topics:

- *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.
- *SYSTP Utility* - Natural utility which provides various TP monitor-specific functions.
- For information on operation and the individual components of Natural in a CICS environment, see the following sections in the *Operations* documentation:
  - ***Node Error Program and Timeout Considerations for Natural***
  - ***CICS 3270 Bridge Support***
  - ***Special Natural CICS Functionality***
  - ***Natural CICS Sample Programs***
  - ***NCIUIDEX User ID Exit Interface***
  - ***Invoking Natural from User Programs***
  - ***Asynchronous Natural Processing under CICS***
  - ***Logging Natural Sessions under CICS***
  - ***Performance Considerations***
  - ***Natural CICS Interface Debugging Facilities***
  - ***Natural Print and Work Files Under CICS***

NCISCPCB Generation Parameters

The Natural CICS Interface system directory is generated by assembling and linking the `NCISCPCB` source module; see the corresponding step in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.

`NCISCPCB` contains the following macros:

- `NCMDIR`
- `NCMTGD`

The purpose of these macros and the individual parameters which can be specified in the macros `NCMDIR` and `NCMTGD` are described in the following sections.

NCMDIR Macro Parameters

The NCMDIR macro is mandatory and must be specified as the first macro in the NCISCPCB source module. It contains various options for the system. The individual parameters which can be specified in the NCMDIR macro are described below.

CICSPLX | ROLLSRV | SIPSERV | SUBSID | TSKEY | TSRECSZ | USERS

CICSPLX - Switching of CICS Application Region

Possible values are:

Value	Explanation
YES	Natural CICS Interface keeps all session relevant data as the Session Information Records (SIRs) and the session data over a pseudo-conversational screen I/O outside of a local CICS Application Owning Region (AOR), thus enabling the switching of CICS AORs. Setting this parameter to YES also requires the profile parameter ADAMODE to be set to greater than 0.
MODE	This setting almost has the same meaning as YES; the only exception is that CICSPLX=MODE allows an ADAMODE=0 profile parameter specification, that is, CICS AOR switching is not possible, but a Natural session may survive the restart of a CICS AOR in an MRO environment.
NO	Vital Natural session data is kept in the local CICS AOR, which in fact disables CICS AOR switching. This is the default value.

Natural PLEX support means that a Natural CICS session removes all its footprints that exist in a CICS application region at CICS task end, as it might never come back into this region. Therefore all Natural CICS session relevant data must be kept outside of a CICS application region, that is, Natural under CICS passes its session information records (SIRs) to the Authorized Services Manager's SIP handler and the session data to the Natural Roll Server at CICS task end. In addition to that, all modules “held”, that is, modules not linked to Natural but directly invoked via standard linkage conventions as RCA modules or the Adabas linkage module, have to be released at CICS task end. It also requires that the restart information is kept in a CICS terminal owning region (TOR) in case of COMARET=YES, or in a CICS data owning region (DOR), which is shared by all participating CICS AORs, in case of COMARET=NO, see the COMARET parameter for details.

If YES or MODE has been specified, and the NCMDIR SUBSID parameter has not been set, the value of the Natural profile parameter SUBSID in effect for the Natural session initializing the NCI environment will be taken.

Caution: Setting this parameter to YES or to MODE automatically sets SIPSERV and the ROLLSRV parameters to YES.

ROLLSRV - Roll Server Rolling

Possible values are:

Value	Explanation
NO	This is the default value, if CICSPLX=NO and SIPSERV=NO. If CICSPLX or SIPSERV is YES, ROLLSRV=YES is forced.
YES	Specifying YES causes Natural CICS Interface to use the *Natural Roll Server* as roll facility only.

If the Natural Roll Server is to be used to save and restore the Natural session data over a screen I/O, this parameter must be set to YES, when the CICSPLX and SIPSERV parameters are both set to NO. If YES has been specified (or forced) and the NCMDIR SUBSID parameter has not been set, the value of the Natural profile parameter SUBSID in effect for the Natural session initializing the NCI environment will be taken.

Note that, for the purposes of Natural CICS Interface, the Natural profile parameter SUBSID is only honored if it is specified dynamically or in the Natural parameter module. It is ignored if it is specified in a parameter string by a profile parameter SYS or PROFILE or in an alternate parameter module (as specified with the profile parameter PARM).

For more information about choosing between the Natural Roll Server and the roll facilities supported by Natural CICS Interface, see ***Choosing between Roll Server and Roll Facilities*** and ***Choosing the Roll Facility***.

SIPSERV - Authorized Services Manager's Session Information Pool

Possible values are:

Value	Explanation
NO	This is the default value, if CICSPLX=NO. If CICSPLX is not NO, SIPSERV=YES is forced.
YES	Causes Natural CICS Interface to keep its session information records (SIRs) in the *Authorized Services Manager*'s session information pool.

With this parameter set or forced to YES, the Natural session information records are kept outside a CICS region, thus enabling Natural to switch a CICS application region after a pseudo-conversational screen I/O.

If YES is specified (or forced) and the NCMDIR SUBSID parameter has not been set, the value of the Natural profile parameter SUBSID in effect for the Natural session initializing the NCI environment will be taken.

Note that, for the purposes of Natural CICS Interface, the Natural profile parameter SUBSID is only honored if it is specified dynamically or in the parameter module. It is ignored if it is specified in a parameter string by a profile parameter SYS or PROFILE or in an alternate parameter module (as specified with the profile parameter PARM).

Caution: If YES is effective for this parameter, the ROLLSRV parameter is forced to YES, unless already specified.

SUBSID - Sub-System ID

Possible values are:

Value	Explanation
xxxx	Defines the sub-system ID for the Natural Roll Server and/or for the Authorized Services Manager.

This parameter defines the Natural sub-system ID to be used for the Natural Roll Server and/or for the *Authorized Services Manager*. If this parameter is not specified, the value of the Natural profile parameter SUBSID will be taken.

Note that, for the purposes of Natural CICS Interface, the Natural profile parameter SUBSID is only honored if it is specified dynamically or in the Natural parameter module. It is ignored if it is specified in a parameter string by a profile parameter SYS or PROFILE or in an alternate parameter module (as specified with the profile parameter PARM).

TSKEY - Prefixes for Natural CICS Temporary Storage Key

This parameter defines the constant prefixes of the temporary storage queues (see explanation below).

This parameter has the same meaning as the TSKEY parameter in the NCIZNEP module (see the Natural *Installation* documentation) and must be specified identically.

Possible values are:

Value	Explanation
(*xxxx*,*yyyy*)	*xxxx* defines the prefix for roll data, whereas *yyyy* defines the prefix for pseudo-conversational restart data.
(NAT2,NCOM)	This is the default value.

When CICS temporary storage (main or auxiliary) is to be used for Natural CICS Interface roll facility or for the communication area for pseudo-conversational Natural tasks (as described with the COMARET parameter of the NTCICSP macro), names for queues of task dependent unique temporary storage must be specified.

These queue names consist of a constant 4-byte key and a task-related key. For terminal-dependent tasks, this task-related key corresponds to the terminal ID, for asynchronous non-terminal tasks it corresponds the CICS unique task number. The constant prefix of the temporary storage queue names is defined by the TSKEY parameter.

Natural CICS Interface requires two 4-byte prefixes: one for roll data and one for pseudo-conversational restart data. *xxxx* defines the prefix for roll data, *yyyy* defines the prefix for pseudo-conversational restart data. The two prefixes must be different from each other and exclusive for Natural under CICS.

When running in a CICSplex environment, the CICS temporary storage prefix for Natural session restart information must be defined in a CICS TST as `REMOTE/SHARED` to be accessible in all participating CICS regions.

TSRECSZ - Record Sizes for Main and Auxiliary Temporary Storage

This parameter defines the maximum record length for rolling of data if CICS temporary storage is to be used as Natural CICS Interface roll facility.

Possible values are:

Value	Explanation
(*nnnnn*,*mmmmm*)	The first subparameter *nnnnn* applies to CICS main temporary storage and must be in the range of `4096` to `32763` or `0` or one of the keywords `MAX`, `YES` or `NO`; ■ if numeric non-zero, this value is used unconditionally; ■ if set to `0` or `NO`, CICS main temporary storage cannot be used for a Natural roll facility; ■ if set to `MAX` or `YES`, a record size of `32763` is taken. The second subparameter *mmmmm* applies to CICS auxiliary temporary storage and must be in the range of `3976` to `32763` or `0` or one of the keywords `MAX`, `YES` or `NO`; ■ if numeric non-zero, this value is used unconditionally; if set to `MAX`, a record size of `32763` is taken; ■ if set to `NO`, CICS auxiliary temporary storage cannot be used for a Natural roll facility; ■ if set to `0` or `YES`, Natural CICS Interface sets the record length which fits into an auxiliary temporary storage control interval, that is, CI size minus VSAM control information minus CICS control information. A user-defined record size greater than CI size results in fewer (logical) roll I/O operations at the expense of additional CICS overhead due to writing spanned records.
`(32748,0)`	This is the default value.

USERS - Session Information Record

This parameter specifies the number of session information record slots (SIRs).

Possible values are:

Value	Explanation
(*nnnnn*, *mmm*)	The subparameter *nnnnn* defines the number of SIRs to be held in the Natural CICS directory module itself. *nnnnn* must be in the range from 1 to 32767. When the SIR slots in the directory are occupied, Natural CICS Interface acquires a CICS shared storage segment, large enough to hold the number of SIRs defined by *mmm*, which must be in the range from 0 to 255. If the subparameter *mmm* is 0 or omitted, the system does not acquire additional storage for SIRs if no free SIR slot is available in the system directory. If so, the Natural CICS system is actually restricted to the number of users specified by the first subparameter. If a value other than 0 is specified for *mmm*, secondary storage segments are allocated automatically as required. Allocated secondary segments are freed again if they are no longer needed.
(100,20)	This is the default value.

Natural CICS Interface permanently holds information about all active Natural sessions. Per session a so-called Session Information Record (SIR) is maintained.

These SIRs are kept

- in a Coupling Facility when running in a z/OS Parallel Sysplex environment;
- in a data space of the *Natural Authorized Services Manager* when running in multiple CICS regions inside a single z/OS system;
- in a CICS region's main storage when running in a single CICS AOR (locally).

However, whenever a Natural session is active in a CICS region, it will occupy a SIR slot in the current application region.

When running locally in a single CICS AOR, the USERS parameter applies to all Natural sessions. When running in a CICSplex environment, USERS applies to the subset of Natural sessions which is currently active in each of the participating CICS AORs.

NCMTGD Macro Parameters

The NCMTGD macro is mandatory and must be specified for each thread group. Natural CICS Interface allows you to define groups of threads. These groups are controlled/chosen by the CICS transaction ID at session initialization. The common thread size for the various groups may differ and the groups can have different options. The thread group definitions are part of the Natural CICS system directory, as they are relevant to the whole system, not just to a single session.

The individual parameters which can be specified in an NCMTGD macro are described below.

PFKEY | PRIMERF | THRDSZE | THREADS | TRAN | TYPE | XTRAN

PFKEY - PF/PA Keys for Thread Group

This parameter defines a single CICS transaction or a list of them.

Possible values are:

Value	Explanation
xxx	Possible values for *xxx* are: PF1 to PF24, PA1 to PA3.
(xxx,xxx,...)	Also a list of keys can be specified. This has to be enclosed in parantheses, for example, PFKEY=(PF12, PF14).

No default value is provided.

When starting a session, Natural CICS Interface scans through all thread group definitions for the current transaction ID, or PF or PA key. If it cannot be found, the first thread group is taken as default.

Caution: At least one transaction ID (in character or hexadecimal format) or one transaction initiating attention identifier must be specified for all groups, except for the first group, which is used as the default group.

PRIMERF - Natural CICS Primary Roll Facility

This parameter defines the Natural CICS Interface primary roll facility for all tasks defined in the associated thread group. Therefore, this parameter does not apply to thread groups with TYPE=NONE.

Possible values are:

Value	Explanation
VSAM	Natural CICS Interface VSAM RRDS roll files are taken as the primary roll facility. If no VSAM RRDS roll file is available in the CICS system, PRIMERF=AUX becomes effective. If the VSAM RRDS roll files become full or is unavailable, the following applies: ■ PRIMERF=AUX becomes effective if auxiliary temporary storage is defined in the CICS system. ■ PRIMERF=MAIN becomes effective if auxiliary temporary storage is not defined in the CICS system.
AUX	CICS auxiliary temporary storage is taken as primary roll facility of Natural CICS Interface. If auxiliary temporary storage is not defined in the CICS system, PRIMERF=MAIN becomes effective.
MAIN	CICS main temporary storage is taken as Natural CICS Interface primary roll facility. The record size is defined by the TSRECSZ parameter. The following applies on z/OS: ■ CICS main temporary storage is memory allocated above the bar. This avoids VSAM I/O activity and communication with a temporary storage server. As a consequence, PRIMERF=MAIN is much faster than PRIMERF=AUX and is therefore recommended if no dynamic transaction routing is used for Natural sessions. However, the roll data is split into records with a maximum size of 32 KB. ■ Natural CICS Interface can also use CICS memory objects as a roll facility. Memory objects are allocated above the bar. However, compared to PRIMERF=MAIN, roll data does not have to be split if it is moved to a memory object. ■ If you want to use memory objects, specify PRIMERF=MAIN and specify MEMOBJR=ON (default setting) with the NTCICSP macro (see the *Parameter Reference* documentation).
NONE	The associated sessions do not roll at all. NONE is not valid for TYPE=SHR groups and for groups with TYPE=ALIAS redefining TYPE=SHR groups. Sessions that are associated with thread groups defined with PRIMERF=NONE cannot be rolled because there is no roll facility to perform this task. These sessions are therefore conversational.

No default value is provided.

This parameter is ignored when using the *Natural Roll Server*; if you force a Natural session with Roll Server to run conversationally with no rolling, value NONE is effective.

For more information about choosing between the Natural Roll Server and the roll facilities supported by Natural CICS Interface, see ***Choosing between Roll Server and Roll Facilities*** and ***Choosing the Roll Facility***.

THRDSZE - Thread Size

This parameter defines the common thread size for TYPE=GETM and TYPE=SHR groups.

Possible values are:

Value	Explanation
nnnnn	The thread size *nnnnn* can be in the range from 40 KB (minimum) to 65532 KB (maximum).

No default value is provided.

Note that this parameter defines the *logical* thread size that is available to Natural. However, Natural CICS Interface adds another 2 KB to the logical thread size for internal administration purposes. This means that the *physical* thread size or length of the thread GETMAIN request is by 2 KB greater than the THRDSZE value.

In case of TYPE=GETM, additional 16 bytes for the heading and trailing CICS storage accounting areas (SAAs) have to be considered.

Important Notes:

1. For GETMAINs of more than 512 KB, CICS aligns these storages at MB boundaries.
2. When using transaction isolation (z/OS only), CICS internally uses 1 MB "pages" in the EUDSA (see the *CICS Performance Guide for details*).

These two facts lead to storage fragmentation and should be kept in mind when setting an appropriate EDSALIM in CICS.

THREADS - Number of Threads or Tasks Per Thread Group

This parameter specifies the number of threads or tasks as described below.

Possible values are:

Value	Explanation
nnn	The number of threads can be equal to 510 or less.

No default value is provided.

For TYPE=SHR thread groups, the THREADS parameter is mandatory and defines the number of threads which are to be allocated via GETMAIN (SVC or SHARED, depending on CICS version) during installation.

For TYPE=GETM and TYPE=NONE thread groups, the THREADS parameter is optional and determines the maximum number of concurrently active Natural tasks per thread group. For these thread

group types, the THREADS parameter does not control storage usage in contrast to TYPE=SHR thread groups (see also *Controlling Storage Usage*).

The number of threads or the number of tasks per thread group is defined by providing thread control blocks (TCBs).

While for TYPE=SHR thread groups, each thread is closely connected to its TCB. Threads are shared by queueing up on the associated TCB. Thread groups of TYPE=GETM and TYPE=NONE only queue up on a TCB to get active.

While sessions with TYPE=SHR thread groups compete for threads, the other session types compete for TCBs with a thread already allocated (TYPE=GETM) or with no allocated thread at all (TYPE=NONE).

When the THREADS parameter is non-zero, the Natural profile parameters DBROLL and MAXROLL and the calls to CMROLL are handled differently for TYPE=GETM/NONE thread groups: As threads cannot be released, the TCB resource held is released, which activates the session with the session data kept in storage.

TRAN - Transaction IDs for Thread Group

This parameter defines a single CICS transaction or a list of them.

Possible values are:

Value	Explanation
(see below)	One or more CICS transaction codes defined in the PCT for Natural.

No default value is provided.

The TRAN parameter expects transaction IDs to be in character format; transaction IDs with non-alphanumeric characters have to be enclosed in apostrophes.

When starting a session, Natural CICS Interface scans through all thread group definitions for the current transaction ID, or PF or PA key. If it cannot be found, the first thread group is taken as default.

A list of transaction IDs has to be enclosed in paranteses, for example, TRAN=(NATU, XYZ).

Caution: At least one transaction ID (in character or hexadecimal format) or one transaction initiating attention identifier must be specified for all groups, except for the first group, which is used as the default group.

TYPE - Thread Type for Group

This parameter defines which type of thread is to be used for a given group.

Possible values are:

Value	Explanation
SHR	CICS shared storage threads are used. The threads available for a thread group are shared by all CICS transactions defined for this group. Thread selection when starting a CICS task is done by an ENQUEUE/DEQUEUE technique. If currently no thread is available, a wait queue for this thread group is maintained. This is the default value. When running in a z/OS Parallel Sysplex environment, the Natural parameter RELO=OFF forces sessions with TYPE=SHR threads to be conversational to prevent a CICS region switch.
GETM	Threads allocated via GETMAIN are used, which means that a thread is actually acquired performing a CICS GETMAIN operation - EXEC CICS GETMAIN FLENGTH - with the thread group's common thread size. Using threads allocated via GETMAIN, each Natural task has exclusive thread storage available until it is terminated; that is, for pseudo-conversational tasks from screen I/O to screen I/O. If the Natural parameter RELO=OFF or PSEUDO=OFF is specified, tasks using threads allocated via GETMAIN are forced to be conversational, as there is no guarantee that after a FREEMAIN of the thread a subsequent GETMAIN obtains the same storage in memory. As thread storage allocated via GETMAIN exclusively belongs to the owning task, however, such tasks can be defined as non-rollable (see the PRIMERF parameter), which means that a given thread belongs to a given task until the end of the Natural session. If so, the task is conversational by design and no rolling is done.
NONE	No threads are used by transactions defined in this thread group and all Natural GETMAIN requests are directly passed to CICS for an EXEC CICS GETMAIN FLENGTH request. By design, such tasks cannot roll and are therefore conversational.
ALIAS	The current NCMTGD macro provides different options for the thread group defined by the previous NCMTGD macro specification. However, only thread groups of TYPE=GETM and TYPE=SHR can be redefined by one or more NCMTGD TYPE=ALIAS macro requests. Up to 99 thread groups are supported, which means that up to 99 NCMTGD macro specifications with TYPE other than ALIAS are recognized.

XTRAN - Hexadecimal Transaction IDs for Thread Group

This parameter is equivalent to the TRAN parameter, but it expects the transaction ID to be in hexadecimal format.

Possible values are:

Value	Explanation
(see below)	Possible values: one or more CICS transaction codes defined in the PCT for Natural.

No default value is provided.

A list of transaction IDs in hexadecimal format has to be enclosed in parantheses, for example, `XTRAN=(D5C1E3E4, E7E8E9)`.

Caution: At least one transaction ID (in character or hexadecimal format) or one transaction initiating attention identifier must be specified for all groups, except for the first group, which is used as the default group.

NTCICSP Macro Parameters

The parameters required for Natural CICS Interface are generated by assembling the Natural parameter module which holds the required NTCICSP macro definitions. The Natural parameter module is created in the corresponding installation step in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.

The NTCICSP macro determines all Natural session options that are relevant in a CICS environment. The individual parameters contained in the NTCICSP macro are described in *CICSP - Environment Parameters for Natural CICS Interface* in the *Parameter Reference* documentation.

6 Customizing VSAM RRDS Roll Files

This chapter describes the customization of VSAM RRDS roll files.

It covers the following topics:

This section does not apply if you are using the Natural Roll Server.

Increasing the Number of VSAM RRDS Roll Files

Up to nine VSAM RRDS roll files can be allocated. Each roll file has an ID consisting of a user-defined prefix followed by a fixed suffix. The prefix can be 1 to 9 characters long. The suffix consists of two characters from `R1` to `R9`.

To add a new VSAM roll file, perform the following steps:

1. Create an empty VSAM RRDS conforming to your local site standards. Then initialize the data set using the batch program `NCISCPRI`, which must have been assembled during the Natural installation.The `SPACE` and `RECORDSIZE` attributes can differ between different roll files, so you can modify them as required to find the best values in your environment.
2. Create an FCT entry and change the CICS JCL accordingly, using the prefix/suffix for both.

The new roll file becomes available when Natural CICS Interface is initialized again.

Decreasing the Number of VSAM RRDS Roll Files

Perform the following steps:

1. Ensure that Natural is not active.
2. Either delete the FCT and JCL definitions or delete the file.

The number of roll files is adjusted when Natural CICS Interface is initialized again.

Changing the Characteristics of the VSAM RRDS Roll Files

Perform the following steps:

1. Execute the procedures described above for decreasing the number of roll files.
2. Execute the procedures for increasing the number of roll files.

7 Natural in CICS MRO Environments

This chapter describes the functionality of Natural in CICS Multi-Region (MRO) Environments.

NTCICSP Parameter COMARET Set to ON

When the `NTCICSP` parameter `COMARET` is set to `ON`, Natural session data are kept in two different CICS regions:

- The session restart information is kept in the COMMAREA linked to the terminal entry in the CICS terminal owning region (TOR).
- The actual session data are kept in the CICS application owning region (AOR); that is, the thread or roll facility.

This may lead to inconsistencies when, for example, the AOR is restarted, but the TOR still contains old "pending" Natural sessions; resuming such a session results in a corresponding error message.

NTCICSP Parameter COMARET Set to OFF

When `COMARET` is set to `OFF`, all Natural session data are kept in the AOR, thus preventing the inconsistencies mentioned above.

However, there may be a security concern when a terminal is removed from the TOR (either back to VTAM or by switching the session manager or power off), and another terminal dialing to this TOR receives the ID of the removed terminal and enters the Natural transaction code: then this terminal resumes the session of the previously removed terminal because of the restart information in the AOR's temporary storage, which contains the terminal ID as part of the queue name.

To prevent such a situation, a node error program (NEP) can be installed (see ***Node Error Program and Timeout Considerations for Natural*** and ***Natural CICS Sample Programs***), which terminates a Natural session when the associated terminal is removed.

8 CICS Node Error Program and Timeout Considerations for Natural

An active Natural session uses CICS resources such as thread storage, roll facility entries (that is, records in a VSAM RRDS file, in a CICS temporary storage queue or main memory) as well as Roll Server slots. These resources are allocated by Natural CICS Interface and correctly released whenever a session under control of Natural CICS Interface terminates normally or abnormally.

If a Natural session termination is not controlled by Natural CICS Interface, a node error program helps to correctly release the acquired resources.

This chapter discusses CICS node error program and timeout considerations.

See also:

- For information on installing a CICS node error program, refer to the corresponding section in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.

Session Termination Not under Control of Natural CICS Interface

In the following situations, Natural CICS Interface does not receive control at session termination and, therefore, cannot properly release the resources still held by the session:

1. A non-Natural program called by Natural issues an `EXEC CICS ABEND CANCEL` command to terminate the CICS task abnormally.
2. Some CICS monitor products offer tools to purge CICS tasks, thus bypassing any abnormal termination exit set by the application.
3. A user disconnects a terminal from the CICS region (by switching the power off or using an adequate session manager function) while the respective Natural session is not active in CICS (pseudo-conversational screen I/O) at that time.
4. A user has been inactive in the Natural terminal session for the period specified with the `TIMEOUT` parameter in the CICS segment of the RACF user profile.

Recovery Mechanisms

Natural CICS Interface provides some recovery mechanisms to recover from such situations.

For example:

Whenever a new Natural session is to be started, Natural checks whether another session with the same terminal ID is active. If such a session exists, it is logically terminated, and all its resources are released prior to starting the new session.

However, if `COMARET=OFF` is set in the `NTCICSP` macro, the information to resume a Natural session is kept in a CICS temporary storage record where the terminal ID is part of the storage queue

name. As a result, another CICS user who tries to start a Natural session with this terminal ID will resume the old session rather than starting a new session. Therefore, we strongly recommend that you set COMARET=ON.

CICS provides the node error program (NEP) exit interface DFHZNEP, which is called when a user disconnects a terminal from the CICS region. The session is lost if the user disconnects the terminal while the respective Natural session is not active in CICS. You can then use DFHZNEP to execute the Natural-specific NCIZNEP node error program and terminate the lost session. This program is delivered with Natural CICS Interface (see also ***Natural CICS Sample Programs***).

If an active and running Natural session is canceled

- using CEMT SET TASK(...) FORCEPURGE
- using CEKL SET TASK(...) FORCEPURGE
- by a CICS monitoring product
- by flushing the Natural session using SYSTP

the respective CICS task is terminated immediately with CICS abend AKEH without NCIZNEP being executed for session cleanup.

NCIZNEP Setup

CICS provides the DFHZNEP default node error program (NEP) that is called when a user disconnects a terminal from the CICS region. Furthermore, CICS provides the DFHZNEPX sample program that is designed for customized error handling in order to invoke user-supplied error processors.

Natural CICS Interface provides the Natural-specific NCIZNEP node error program. You can use the XNCINEP2 sample node error program in order to set up DFHZNEP and execute NCIZNEP to terminate a lost session. XNCINEP2 can be used instead of the sample program DFHZNEPX.

XNCINEP2 performs the following functions:

- Enables an error processor to receive control for all possible error codes passed to DFHZNEP. If XNCINEP2 receives control, it issues a CICS trace request that shows with which error codes DFHZNEP has been invoked on certain actions and related TCTTE error information.
- Calls the NCIZNEP module via EXEC CICS LINK which performs a Natural session cleanup.

The XNCINEP2 sample program is supplied in the source library of Natural CICS Interface. If DFHZNEPX has already been customized, copy the relevant source code from XNCINEP2 into DFHZNEPX and insert it into an appropriate position before the trailing EXEC CICS RETURN call in DFHZNEPX, considering other products or components for which error processing is performed.

NCIZNEP Functionality

NCIZNEP tries to resume a session asynchronously and subsequently terminate it logically based on the session restart information (NEXTTRANSID and restart data in COMMAREA or CICS temporary storage) of a terminal session with pending pseudo-conversational screen I/O.

Upon completion, NCIZNEP cleans the parameter input indicating to the caller (usually DFHZNEP) whether it has successfully completed its work and launches the cleanup task for the Natural session.

If more than one Natural CICS Interface version is active in a CICS system, cleanup processing performed by the called Natural version-specific node error program can fail, because the Natural session to be terminated is hosted by a different Natural CICS Interface version. Upon execution completion of NCIZNEP, DFHZNEP can test whether the operation was successful. If the operation was not successful, NCIZNEP will call another node error program related to a different Natural CICS Interface version.

If a user disconnects a terminal from the CICS region while the respective Natural session is not active in CICS, DFHZNEP can receive control more than once for various internal error codes, since each internal error code is related to a specific CICS error message.

If you want NCIZNEP to attempt to immediately purge a currently active Natural session for which NCIZNEP receives control, specify PURGE=YES for the NCIZNEP module. See the installation procedure for Natural CICS Interface and the section *CICS Startup Parameters* in *Installing Natural CICS Interface on z/OS* in the *Natural Installation* documentation.

Requirements for MRO/CICSPlex Environments

In general, the following is required for MRO/CICSPlex environments:

- DFHZNEP and NCIZNEP must be installed in the TOR.
- The NCIZNEP module must be defined in CICS with EXECKEY(CICS).

For MRO/CICSPlex environments where you want NCIZNEP to purge a currently active Natural session, consider the following additional requirements:

- The NCIZNEP module must also be defined in the CICS AORs associated with a TOR. A Natural session can only be purged in the AOR where a transaction is started to purge the task.
- A transaction ID must be defined for the NCIZNEP module in the CICS TOR and the AORs with the attributes TASKDATAKEY(CICS) and PRIORITY(255)

- The transaction ID must also be specified with the NEPTRAN parameter in the CICS startup parameters for the NCIZNEP module as described in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.
- The Natural transactions for which you want to purge currently active Natural sessions have to be defined in CICS with the attribute SPURGE(YES).

Special Considerations for Adabas System Coordinator

If you want to call the node error programs NCIZNEP (for Natural CICS Interface) and CORNEP (for Adabas System Coordinator) from DFHZNEP, you should consider that both NCIZNEP and CORNEP may attempt to release Adabas-related resources at the same time. In order to avoid interferences due to concurrent execution, always call NCIZNEP before CORNEP and slightly delay the start time of CORNEP to make sure that NCIZNEP has completed before CORNEP starts executing.

For more information, see the *Adabas System Coordinator Installation* documentation.

Timeout Handling

If a user session is timed out, the resources held by the Natural session are not released by CICS. A user session is timed out, for example, if the user has been inactive in the Natural terminal session for the period specified with the TIMEOUT parameter in the CICS segment of the RACF user profile.

In the case of a timeout due to the setting of the TIMEOUT parameter in the RACF user profile, the action that CICS performs is determined by the value of the GNTRAN system initialization parameter. Natural resources are not properly released if either GNTRAN=NO is set or the executed "good night" program does not call the NCIZNEP module.

To make sure that all resources held by Natural are properly released

1. For the GNTRAN parameter, specify a CICS transaction ID for which the PROGRAM attribute specifies XNCIGNIT.
2. Use the XNCIGNIT sample node error program supplied in the source library of Natural CICS Interface (see also ***Natural CICS Sample Programs***) and define it in CICS.

   XNCIGNIT must be assembled and linked with the CICS EXEC interface modules DFHELII and DFHEAI0.

   XNCIGNIT calls the NCIZNEP module via EXEC CICS LINK, which performs a Natural session cleanup.

You must perform the following for MRO/CICSPlex environments:

- In the TOR, define and install the XNCIGNIT sample program and the CICS transaction ID for which the PROGRAM attribute specifies XNCIGNIT.
- In the XNCIGNIT sample node error program, add the SYSID parameter to the EXEC CICS LINK statement that executes NCIZNEP. SYSID must specify the system name of an AOR that relates to the TOR. You can specify any AOR, because a CICS task is not active for a session in an AOR when the session is timed out.

API USR4006N for Session Retrieval or Cancellation

The USR4006N application programming interface (API) allows you to retrieve or cancel Natural sessions based on selection criteria, such as the period of inactivity of terminal sessions.

It can happen that all sessions are read sequentially to find one or more sessions that match a specified criterion because USR4006N is not able to randomly select sessions by a specific criterion. Therefore, we recommend that you execute USR4006N only occasionally, and only if you want to remove all Natural sessions that satisfy a specific criterion in a single run. Alternatively, you can use the NCIZNEP module and the XNCIGNIT sample node error program which operate on a single Natural session only and need not access all sessions.

9 CICS 3270 Bridge Support

This chapter describes CICS 3270 Bridge support.

Default Support of CICS 3270 Bridge

By default, Natural CICS Interface supports the CICS 3270 Bridge by being able to deal with "bridged devices", that is, terminals which are emulated via a CICS 3270 bridge exit.

Full CICS 3270 Bridge Support

If you want full CICS 3270 Bridge support, you have to install the NCI load module `NCIXFATU`. Refer to the corresponding installation step in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.

NCIXFATU - NCI Load Module

The `NCIXFATU` module is a CICS `XFAINTU` Global User Exit (GLUE). Its purpose is to release Natural resources in case the bridge facility's keep-time has expired and an associated Natural session has not been terminated yet.

The `NCIXFATU` module calls the `NCIZNEP` module through `EXEC CICS LINK`, which performs a Natural session cleanup.

Profile Parameter DSC=OFF Recommended

When you are using the CICS 3270 Bridge, you are recommended to start a Natural session with profile parameter `DSC=OFF` (data-stream compression for 3270-type terminals disabled) to force Natural always to send full screens rather than the delta to the previous screen.

10 Threadsafe Considerations

Since Natural is fully reentrant, Natural CICS Interface can be defined as threadsafe by using the CICS `CONCURRENCY(REQUIRED)` attribute. Natural CICS Interface can then execute in the CICS open transaction environment (OTE) under an open TCB to reduce the QR CPU constraint by moving tasks to other processors.

Natural CICS Interface provides for extra serialization using CICS ENQ/DEQ when running under an open TCB.

In order to minimize these serialization efforts, it is highly recommended

- to use `TYPE=GETM` threads without the `THREADS` parameter specified (or `THREADS=0`),
- to use the *Natural Roll Server* rather than roll facilities in CICS.

Notes:

1. All user programs defined as `CSTATIC` have to be threadsafe.
2. All dynamic user programs have to be threadsafe if they are invoked using standard linkage conventions either explicitly (that is, using the terminal command `%P=S`, `%P=SC`, `*P=L` or `%P=LS`, or the `PGP` profile parameter with the respective property) or implicitly (that is, when the `NTCICSP` macro parameter `SLCALL` is set to `ON`). If these programs are not threadsafe, you can use the `%P=SQ` terminal command or `PGP` profile parameter (with the `STDLQ` property set) to call quasi-reentrant user programs. Additional information on threadsafe programs can be found in the IBM documentation related to CICS application programming.

11 CICS Open Transaction Environment Considerations

If you want to execute Natural in the CICS open transaction environment (OTE) under an open TCB, all external subprograms must be threadsafe and fully reentrant if they are called by Natural using standard linkage conventions (direct branch using a BASR instruction) instead of an `EXEC CICS LINK` command. It is insufficient that the called subprograms are quasi-reentrant only. This applies to all external subprograms defined to Natural with the `CSTATIC`, `RCA` or `PGP` profile parameter, or called after the terminal command `%P=S` was issued.

If a subprogram is quasi-reentrant but not threadsafe, use the `%P=SQ` terminal command instead of `%P=S`, or use the `PGP` profile parameter with the `STDLQ` property set. When using the `%P=SQ` terminal command or `PGP` profile parameter (with `STDLQ`), the external subprogram is executed on the CICS QR TCB rather than an open TCB. Natural then issues the CICS command `EXEC CICS LINK PROGRAM('NCILINKQ')` to switch to the QR TCB and execute the NCILINKQ program which calls the subprogram by using standard linkage conventions.

If you use the `%P=SQ` terminal command or `PGP` profile parameter (with `STDLQ`), define NCILINKQ as described in *Routing Module for Quasi-Reentrant Standard Linkage Calls (%P=SQ)* in the section *Installing Natural CICS Interface* in the *Installation for z/OS* documentation.

If a called external subprogram is not defined to Natural with the `CSTATIC` or `RCA` profile parameter but has a PPT entry in the operating PPT and resides in a load module library so that CICS can locate the subprogram and load it, the call is accomplished by an `EXEC CICS LINK` command. If the called subprogram is defined with `CONCURRENCY(QUASIRENT)`, CICS automatically switches to the QR TCB to execute the call.

If a called external subprogram is defined to Natural with the `CSTATIC` or `RCA` profile parameter, the `%P=SQ` terminal command and the `PGP` profile parameter (with `STDLQ`) do not apply.

In order to avoid problems, use the `API(OPENAPI)` and `CONCURRENCY(REQUIRED)` program attributes consistently for all programs calls that use a direct branch (BASR) instruction. This also applies to front-end programs of Natural add-on products that call the environment-dependent nucleus of Natural. For further information, see ***Threadsafe Considerations***.

12 Support for CICS Channels and Containers

The IBM CICS Transaction Server for z/OS supports channels and containers for `EXEC CICS LINK`. In this respect CICS containers can be considered as named COMMAREAs without the 32 KB limit.

Natural CICS Interface supports CICS containers in two ways:

1. Via a special `SET CONTROL 'P=CC'`, the `CALL` statement parameter data is passed in a container.
2. When the `NTCICSP` macro parameter `CNTCALL` is set to `ON`, a `%P=C CALL` automatically uses a CICS container rather than a CICS COMMAREA, when the parameter data passed with the `CALL` statement exceeds 32 KB.

In both cases, the default container name is `NCI-COMMAREA` unless a container name is defined explicitly via application programming interface `USR4204N` prior to the “real” `CALL` statement.

13 IBM Language Environment (LE) and Natural CICS Interface

Natural CICS Interface supports LE programs. This document contains information on LE enablement of Natural under CICS.

CICS Transaction Server for z/OS - LE-compliant

If supported by the CICS Transaction Server for z/OS installed at your site, Natural CICS Interface is LE-compliant by itself, that is, a Natural CICS Interface task can directly `CALL` (standard linkage conventions, not CICS LINK) LE programs written in languages such as C, COBOL or PL/I, when

- `SET CONTROL 'P=LS'` has been specified,
- `SET CONTROL 'P=S'` has been specified,
- `NTCICSP` macro parameter `SLCALL=ON` has been specified and the program to be called is *not* a CICS program.

14 Special Natural CICS Functionality

This chapter explains special Natural CICS functionality.

Calling Non-Natural Programs

One of the first actions a Natural task does at its start, is to activate an exit for abnormal termination processing. This exit is used to release all resources including the thread in the case of an abnormal termination. Therefore, a non-Natural program must not issue `EXEC CICS ABEND CANCEL` or the equivalent macro level request, as such a request cancels the current session ignoring any active exit. If so, Natural is not able to clean up its resources, and the thread and the roll facility are not released.

A thread is assigned to a Natural task whenever a Natural program is active. This is also true when non-Natural programs are called (following CICS linkage conventions).

Therefore, such programs should not do excessive I/O and other work load without Natural receiving control in between. If a non-Natural program is doing conversational screen I/O, you can code a `SET CONTROL 'P=V'` statement in the Natural program that calls the non-Natural program before the calling statement: this indicates that parameter data are copied out of the thread and the session is rolled out before calling the non-Natural program.

Calling Non-Natural Programs via Standard Linkage Conventions

A non-Natural program is invoked (CALLed) by Natural in the way programs are invoked within the underlying operating and/or TP monitor system.

In CICS, non-Natural programs are invoked by means of `EXEC CICS LINK` requests. However, when, for example, the same subroutine program (not issuing any CICS or operating system request) is to be used for both batch and online processing, a non-Natural program may also be invoked by using standard linkage conventions, that is, via `BASR R14,R15`.

For further information, see the terminal command `%P=S` in the *Terminal Commands* documentation. See also the parameter `SLCALL` in the macro `NTCICSP`.

Calling Non-Natural Programs with Parameter Values in a COMMAREA or in a Container

By default, non-Natural programs are called with the addresses of the request parameter lists (see the description of the `CALL` statement in the Natural *Statements* documentation) passed in the TWA and/or a COMMAREA (depending on the setting of the `NTCICSP` macro parameter `CALLRPL`).

A more CICS-like method is to pass the parameter values in a CICS COMMAREA or a CICS Container (see ***Support for CICS Channels and Containers***), particularly when the called program resides in another CICS region - Distributed Program Link (DPL) -, as addresses within the “calling” region are not accessible by the “called” region.

For details and restrictions, see the terminal commands %P=C and %P=CC in the *Terminal Commands* documentation.

Prerequisite:

This functionality requires CALLRPL to be set to ON in NTCICSP.

When the parameter values are passed in a CICS COMMAREA or CICS container, no parameter list pointers are passed in the CICS TWA, regardless of the CALLRPL setting.

Dummy Screen I/O with Natural under CICS

If a SET CONTROL 'QO' statement is placed before a Natural statement that causes a screen I/O, the terminal output is not executed by Natural under CICS. Consequently, both the Enter key and user input are not passed back to Natural.

This functionality may be useful in the following situations:

1. When leaving pseudo-conversational screen I/O to non-Natural programs called by Natural. The non-Natural program performs the EXEC CICS SEND operation and returns to Natural. Due to the SET CONTROL 'QO' statement, the next Natural screen I/O terminates the task of a pseudo-conversational session. Upon screen input, Natural receives control and invokes the non-Natural program again, which then performs the EXEC CICS RECEIVE.
2. When changing the Natural pseudo-conversational transaction ID "in-flight" without requiring a terminal operator intervention:

```
MOVE *INIT-ID TO termid
CALLNAT 'CMTRNSET'  trnid  /* change the restart transaction ID

* starting a task on your terminal forces an interrupt as if
* pressing any attention identifier

CALL 'CMTASK' USING trnid
H'0000' H'00000000' termid
SET CONTROL 'QO'
INPUT 'DUMMY'                /* dummy I/O, which you will never see
WRITE 'HELLO' *INIT-PROGRAM  /* now the new transaction ID is active ↩
```

3. When switching to an application outside Natural, perhaps even in another CICS AOR (application-owning region), without requiring a terminal operator intervention:

```
* starting a task on your terminal forces an interrupt as if
* pressing any attention identifier

CALL 'CMTASK' USING trnid data-length start-data termid
SET CONTROL 'QO'
INPUT 'DUMMY'                  /* dummy I/O, which you will never see
WRITE 'HELLO' *INIT-PROGRAM    /* now the new transaction ID is active
```

In this case, it is the responsibility of the application being invoked to stack the Natural restart data when they are passed in a CICS COMMAREA, as a COMMAREA most likely is used by the new (pseudo-conversational) application, too.

15 Natural CICS Sample Programs

This chapter contains an overview of the Natural CICS sample programs.

Sample Programs in Natural CICS Source Library

The following sample programs are supplied in the Natural CICS source library:

Front-End Programs

Name	Language	Function
XNCIFRNP	Assembler	Initalization program that initializes the Natural CICS environment at CICS start-up.
XNCIFRNL	Assembler	Front-end program for invoking Natural via EXEC CICS LINK.
XNCIFRNR	Assembler	Front-end program for invoking Natural via EXEC CICS RETURN IMMEDIATE.
XNCIFRNS	Assembler	Front-end program for invoking Natural via EXEC CICS START.
XNCIFRNX	Assembler	Front-end program for invoking Natural via EXEC CICS XCTL.
XNCIFRNN	Assembler	Front-end program for invoking Natural via EXEC CICS LINK without front-end parameters.
XNCIFRCL	COBOL	Front-end program for invoking Natural via EXEC CICS LINK.
XNCIFRCN	COBOL	This is a dummy front-end program for invoking Natural via EXEC CICS LINK for LE compliance.
XNCIFRCR	COBOL	Front-end program for invoking Natural via EXEC CICS RETURN IMMEDIATE.
XNCIFRCS	COBOL	Front-end program for invoking Natural via EXEC CICS START.
XNCIFRCX	COBOL	Front-end program for invoking Natural via EXEC CICS XCTL.
XNCIFRPL	PL/1	Front-end program for invoking Natural via EXEC CICS LINK.
XNCIFRPN	PL/1	This is a dummy front-end program for invoking Natural via EXEC CICS LINK for LE compliance.
XNCIFRPR	PL/1	Front-end program for invoking Natural via EXEC CICS RETURN IMMEDIATE.
XNCIFRPS	PL/1	Front-end program for invoking Natural via EXEC CICS START.
XNCIFRPX	PL/1	Front-end program for invoking Natural via EXEC CICS XCTL.
XNCIFRDN	C	This is the dummy front-end program for invoking Natural via EXEC CICS LOAD and BASR for LE compliance.

Back-End Programs

Name	Language	Function
XNCIBACK	Assembler	Termination Data Dump: This back-end program displays the Natural termination message and any termination data in dump format. If invoked from an asynchronous task, the Natural termination message will be issued on the operator console, and potential termination data will be dumped. NCIBACK can also be invoked by a back-end transaction (RET=*xxxx* or RTI=*xxxx* or STR=*xxxx*, where *xxxx* is the transaction code associated with XNCIBACK).

User Exits

Name	Language	Function
XNCIDIRX	Assembler	System Directory Module Name Exit: This source module contains a sample system directory module name exit (see also ***NCIDIREX - System Directory Module Name Exit Interface***).
XNCIDTPX	Assembler	DTP Terminal Exit: This source module contains a sample DTP terminal exit (see also ***NCIDTPEX - DTP Terminal I/O Exit Interface***).
XNCIRDC1	Assembler	Exit for SYSRDC: This program provides a sample exit for the SYSRDC utility; see the relevant section in the *Utilities* documentation.
XNCITIDX	Assembler	Terminal ID Exit: This program provides a sample user exit to test the terminal ID and/or to set a logical terminal or session ID (see also ***NCITIDEX - Terminal ID Exit Interface***).
XNCITIOX	Assembler	DTP Terminal Exit: This source module contains a terminal I/O exit that is more general than the XNCIDTPX sample (see also ***NCIDTPEX - DTP Terminal I/O Exit Interface***).
XNCIUIDX	Assembler	User ID Exit: This program provides a sample user exit to test/set the user ID (see also ***NCIUIDEX User ID Exit Interface***).
XNCIXIDX	Assembler	Transaction ID Exit: This program provides a sample user exit to test/set the pseudo-conversational transaction ID (see also ***NCIXIDEX Transaction ID Exit Interface***).

Subprogram Calls

Name	Language	Function
XNCI3GC1	COBOL	This program provides a sample COBOL call to a Natural subprogram under CICS.
XNCI3GC2	COBOL	This program provides a sample COBOL call to a Natural subprogram under CICS.
XNCI3GC3	COBOL	This program provides a sample COBOL call to a Natural subprogram under CICS.
XNCI3GP1	PL/1	This program provides a sample PL/1 call to a Natural subprogram under CICS.
XNCI3GP2	PL/1	This program provides a sample PL/1 call to a Natural subprogram under CICS.
XNCI3GP3	PL/1	This program provides a sample PL/1 call to a Natural subprogram under CICS

Other Programs

Name	Language	Function
XNCIUCTR	Assembler	Upper/lower case switch: This program serves to switch the terminal into upper/lower case mode.
XNCINEP2	Assembler	This node error program calls the NCIZNEP module.
XNCIGNIT	Assembler	"Good night" program: This sample program calls the NCIZNEP module for Natural session cleanup.
XNCITRPX	Assembler	Debugging aid: If necessary for error diagnosis, Software AG technical support personnel will advise you of the usage of this program.

16 Invoking Natural from User Programs

This chapter describes the various ways of how Natural can be invoked from user programs.

Commands for Activating a Natural Session

This section covers the following topics

A Natural session can be activated by user front-end programs with one of the following commands:

- `EXEC CICS XCTL`
- `EXEC CICS LINK`
- `EXEC CICS RETURN IMMEDIATE`
- `EXEC CICS START`

or the equivalent CICS macro level requests.

Using EXEC CICS XCTL or EXEC CICS LINK

When using `EXEC CICS XCTL/LINK`, the parameters used by Natural can be passed in a CICS COMMAREA or in the TWA.

- Natural determines the location of the startup parameters by inspecting the length of the COMMAREA passed to it during session initialization.
- If the length is 22, Natural tries to locate the parameters in the COMMAREA, otherwise it is assumed that they have been passed in the TWA.

To identify a front-end program properly, it is mandatory that the first 4 bytes of the front-end parameter list represent the current transaction ID.

The transaction ID associated with the front-end program must have a TWA size that is equal to or greater than the Natural TWA size; see also *ncitransact* in *Installing Natural CICS Interface on z/OS* in the Natural *Installation* documentation.

Using EXEC CICS RETURN IMMEDIATE

When using `EXEC CICS RETURN IMMEDIATE`, the front-end parameters used by Natural can be passed in a CICS COMMAREA and the dynamic parameters used by Natural can be passed with `INPUTMSG (...)` and `INPUTMSGLEN (...)` of the `EXEC CICS RETURN IMMEDIATE` command.

Using EXEC CICS START

When using `EXEC CICS START`, the front-end and dynamic parameters used by Natural can be passed with `FROM (...)` and `LENGTH (...)` of the `EXEC CICS START` command. The parameters are described on the following page.

Using Distributed Program Link (DPL)

When using `EXEC CICS LINK` with the `SYSID` parameter pointing to a remote region, the front-end and dynamic parameters used by Natural have to be passed in a CICS COMMAREA. In addition also a `TRANSID` parameter has to be specified naming the transaction code of a mirror transaction with a TWA size satisfying Natural's requirements.

It should be noted that the same restrictions as for asynchronous Natural sessions apply to Natural sessions invoked via DPL, that is, no input is possible (therefor the same dynamic parameter settings are recommended) and the caller just gets control back after Natural session termination.

Sample Programs

A series of sample programs for the various programming techniques is supplied in the Natural CICS source library; see also *Natural CICS Sample Programs*

Using the External Subroutine CMTASK

It is possible to start a Natural session from a Natural program by calling the external subroutine `CMTASK`. Refer to the sample Natural program `ASYNCICS` in library `SYSEXTP`.

Front-End Parameters

The following list of parameters must be supplied to invoke Natural from a user front-end program:

Pos.	Contents
1 - 4	**Invoking transaction ID** This value must be equal to the current transaction ID. Via the invoking transaction ID, Natural identifies that it was called by a user front-end program. When being called with XCTL, the transaction is restarted at the end of the Natural session via RETURN with TRANS ID, unless a return program name is specified (see 5th parameter).
5 - 8	**Address/offset of dynamic parameter string** If dynamic parameter overwrites are to be evaluated, this value should be set to the address located 12 bytes before the dynamic parameter assignment string. When being called with START or DPL, the field must be set to the offset (relative to the start of the front-end parameter list) of the address located 12 bytes before the dynamic parameter assignment string.
9 - 10	**Length of the dynamic parameter string** Zero indicates that no parameters are to be passed. 32760 is the maximum length allowed. If the maximum value is exceeded, the session is terminated with a corresponding error message.
11 - 14	**Natural transaction ID** The value specified is the transaction ID to be used for controlling a pseudo-conversational Natural session, when being called with START or XCTL. This transaction is invoked each time the Natural session is restarted in pseudo-conversational mode; that is, with each terminal I/O. If the Natural transaction ID is not specified, Natural restarts the transaction ID which initiated the current CICS task, and the front-end program regains control after each pseudo-conversational I/O.
15 - 22	**Back-end program name** This 8-byte value is the program name to which control is transferred at the end of the Natural session with a CICS XCTL command, rather than restarting the calling transaction ID via RETURN with TRANSID. If this field is numeric in the first byte, Natural simply RETURNs without activating any back-end. Please note that this field can be superseded by the Natural profile parameter PROGRAM. For the conventions of calling non-Natural back-end programs, refer to the Natural Operations documentation.

Front-End Invoked via LINK

On return to the front-end, Natural indicates in the TWA if the session has terminated or not: when the session has terminated, the TWA holds regular back-end information (see Back-end Program Calling Conventions in the *Operations* documentation), else Natural puts the `NEXTTRANSID` into the first four bytes of the TWA.

If Natural is running in pseudo-conversational mode (profile parameter `PSEUDO` set to `ON`) and has been invoked by `EXEC CICS LINK` (or the equivalent CICS macro level request), the original invoking transaction is invoked each time Natural writes to a terminal and waits for input, which means that Natural issues a "logical" `CICS RETURN TRANSID (..)` after having written its restart information into CICS temporary storage.

The invoking transaction must recognize this situation (for example, by checking whether a `NEXTTRANSID` has been sent or by the existence of `NCOM`*`xxxx`* TS records - where `NCOM` is the Natural CICS parameter generation option and *`xxxx`* is the terminal ID -) and pass control back to Natural.

The advantage of this method is that, during the session, the front-end program can decide to pass control to another application (for example, COBOL) and to resume the Natural session later.

For further details see the `PSEUDO` parameter description in the *Parameter Reference* documentation.

Per design, Natural treats a LINK front-end program as a back-end program at session termination, that is, the *Back-End Program Calling Conventions* apply.

In CICSplex Environments

Make sure that the `NCOM`*`xxxx`* TS records can be accessed by all participating CICS AORs (for example via appropriate CICS TST definitions).

Alternatively the LINK front-end program can also pick up the NCI session restart information in CICS temporary storage on task termination and pass it in a CICS COMMAREA by itself; such a COMMAREA has then to be put into CICS temporary storage again prior to invoking Natural for session resume.

Front-End Invoked via RETURN IMMEDIATE

This front-end technique only works for Natural terminal sessions. Natural scans for start-up parameters supplied with the COMMAREA. Note that when using this technique, potential dynamic parameters cannot be passed chained to the front-end parameters, that is, the dynamic parameters' address fields must be zero. Instead, potential dynamic parameters can be passed via terminal input data, which are obtained by Natural by an `EXEC CICS RECEIVE` command.

Front-End Invoked via START

If the Natural session is a started task (that is, invoked by an `EXEC CICS START` or `EXEC CICS LINK/XCTL` command by a front-end user program which has been `START`ed), Natural first scans for startup parameters supplied with the COMMAREA, then it scans for parameters in the TWA and finally it tries to obtain the necessary parameters by an `EXEC CICS RETRIEVE` command.

Front-End Invoked via XCTL

If the Natural session is initiated from a front-end program with `XCTL` and no return program is specified (that is, neither a fifth parameter in the session startup parameters nor a `PROGRAM` specification in the Natural dynamic parameters or the `NTPRM` macro), Natural restarts the user front-end transaction at session termination via `RETURN` with `TRANSID` by internally simulating a `PROGRAM='RET=`*`xxx`*`'` specification, with *xxx* being the front-end transaction code.

To avoid a loop condition, logic must be included into the user front-end routine to decide whether a new session is to be started or an old session is to be resumed.

Front-End Invoked via Distributed Program Link (DPL)

If the Natural session is invoked via DPL, Natural first determines if it is directly invoked in the server region or indirectly via `EXEC CICS LINK/XCTL` by a local front-end program. When being invoked directly, Natural retrieves the start parameters from the CICS COMMAREA. When being invoked indirectly, Natural scans for startup parameters supplied with the COMMAREA, then it scans for parameters in the TWA. On return Natural passes regular back-end data in the TWA when there is a local LINK front-end program available, otherwise it returns the termination message and potential back-end data in the remote client's COMMAREA.

Invoking Front-End Program as Back-End

If the Natural session is initiated from a front-end program and this program is also specified to be the return program, the user front-end should also check for the initiating transaction ID.

In particular this applies if the front-end program is not in pseudo-conversational mode but Natural is in conversational mode.

In this case Natural is invoked again rather than getting terminated, but this time without detecting that it is called by a front-end program, as the first parameter in the startup parameters is the Natural transaction ID.

17 Asynchronous Natural Processing under CICS

This chapter contains special considerations that apply when you are using asynchronous Natural under CICS.

Asynchronous Natural Processing

Asynchronous Natural processing is generally discussed in the section *Asynchronous Processing* in the *Operations* documentation; however, some additional considerations apply when running under CICS. These are described in the following sections.

Asynchronous Natural Sessions under CICS

Make sure that appropriate SENDER and OUTDEST destinations are specified for an asynchronous Natural session; otherwise, any output (for example, unexpected error messages) will lead to an abnormal termination.

Also, make sure that a suitable message switching transaction ID (MSGTRAN) is specified in the NTCICSP macro of the Natural parameter module and defined in CICS.

In addition to CICS terminal IDs and transient data destinations for SENDER and OUTDEST, the following keywords are supported by Natural CICS Interface:

DUMMY	Any output is ignored.
CONSOLE	Any output is routed to the operator console. When dealing with the console, the terminal type should be switched accordingly, using the profile parameter TTYPE or the terminal command %T= set to ASYL or other.

By default, the 3270 data stream protocol is used for output of an asynchronous Natural session under CICS.

It is also possible to send Natural output data without any 3270 terminal or printer control information to, for example, a CICS message destination such as CSSL. This can be accomplished by switching into line mode using a SET CONTROL 'T=' statement or by starting with profile parameter TTYPE=*xxxx*, where *xxxx* is BTCH or ASYL. All Natural output is then sent line by line, with a leading ASA control character when the Natural profile parameter EJ is set to ON; with EJ=OFF, no control character is sent at all.

Caution: When SET CONTROL 'T=*xxxx*' or SET CONTROL '+' is used, or when personal-computer support is enabled (profile parameter PC set to ON), the Natural system variable *DEVICE will be modified, which means that it can no longer be used to determine an asynchronous Natural session.

Note that some parameter settings for asynchronous Natural sessions can be forced by setting the NTCICSP macro parameter RCVASYN to ON.

Testing and Debugging

Recent CICS versions offer the transaction CEDX which enables debugging of asynchronous tasks in CICS. In earlier CICS versions, such debugging was only possible with terminal tasks.

If you want to test asynchronous Natural sessions without CEDX, start the asynchronous Natural session from a terminal, and either specify ASYN as the very first five characters in the dynamic parameter string, or specify the profile parameter TTYPE=ASYN or TTYPE=ASYL. Natural CICS Interface then sets up an asynchronous Natural session. Note that even though Natural handles the session as asynchronous, CICS keeps on treating it as a terminal session.

18 Logging Natural Sessions under CICS

This chapter describes how information about Natural sessions can be logged in a file which can be processed and evaluated in batch mode.

Logging Facility

Optionally, information about Natural sessions can be logged in a file which can be processed and evaluated in batch mode.

In contrast to the online *SYSTP Utility*, which just gives a snap shot of the current system usage, this logging facility can be used to keep track of the Natural CICS system usage over a longer period of time.

Special Considerations

- It is possible that several Natural CICS environments (that is, several system directories with unique threads, roll facilities, and buffer pools) share the same Natural log destination. When an SCP environment is initialized, a "system ID" is written into the system directory. This system ID is part of an evaluation program to "sort" log records by Natural CICS system environment.
- You are recommended to define the Natural log file in the Natural application CICS, as logging to a "remote" log file would degrade performance.
- When running the log file evaluation program (see *SYSTP in Batch Mode* in the Natural *Utilities* documentation), the log file should be closed in CICS, otherwise unpredictable results may happen due to the last buffer being still in storage or the EOF record missing on file.
- Sufficient disk space should be reserved for the Natural log file; preferably the log file should be defined using secondary allocation.

Natural Log File Definition

The Natural log file is a sequential disk file; that is, an "extra partition destination" in terms of CICS. By default, the internal (logical) name of the log file is `NLOG`; this name can be changed by specifying the `LOGDEST` parameter in the `NTCICSP` macro.

The log file has to be defined in a CICS DCT as `TYPE=EXTRA` with associated data set control information (`TYPE=SDSCI` entry in DCT). This file must also be defined in the CICS start-up JCL (DD statement).

Natural Log Records

The following records are logged in the Natural log file:

Natural CICS System Restart Record

Length=96

After successful SCP system initialization, a record that holds the initialization date and time as well as other system data like the common system prefix, the number of RCBs or the number of thread groups, is written to the log file.

When this first log request fails, the Natural log file is flagged in the system directory as not available and no further logging takes place.

System restart records are written whenever the system highwater marks are reset by the corresponding system administration function of the SYSTP utility (see the Natural *Utilities* documentation). In addition to the system start information, these records contain the terminal ID and the user ID of the SYSTP user.

Natural Session Termination Record

Length=216

On (normal or abnormal) termination of a Natural session, a session log record is written to the log file. This record is internally split into six parts:

1. The record control part which holds the actual session statistics:
   - the current date and time (that is, the date and time when the session terminated),
   - the system ID which indicates the Natural CICS environment in which the session was active,
   - the record type = session record.

   The record control part is common to all Natural log records to distinguish the different record types. Macro NCMLOG holds the record layouts.
2. The user session part which holds the actual session statistics:
   - the terminal ID,
   - the (last) user ID,
   - the session start date and time,
   - the maximum storage allocated by the session,

- the number of session resumes/roll ins,
- the maximum number of records rolled by the session (if any).

3. The thread group part which holds the current data of the thread group associated with the session:
   - the thread group number,
   - the number of TCB slots in the group (if any),
   - the common thread size of the group,
   - the maximum storage allocated in the group by any session,
   - the maximum number of sessions active in the group,
   - the maximum wait queue size of the group (with `TYPE=SHR` thread groups) and the maximum number of sessions concurrently active in the group (with `TYPE=GETM` thread groups),
   - the number of times this maximum wait queue size was reached.
4. The thread part which holds the data of the `TYPE=SHR` thread used as last thread by the session (if used at all):
   - the thread name,
   - the thread use count,
   - the highest thread storage used by any session,
   - the number of session resumes/roll-ins into this thread,
   - the maximum wait queue size of this thread,
   - the number of times this maximum wait queue size was reached.
5. The roll facility part which holds information about the roll facility to which the session was assigned (if it was at all):
   - the roll facility name,
   - the maximum number of sessions assigned to this roll facility,
   - the record size of the roll facility,
   - the slot size of the roll facility,
   - the number of slots in this roll facility,
   - the maximum number of roll-outs to / roll-ins from this roll facility.
6. The system directory part which holds statistics about the global system usage:
   - the maximum number of UCB block extensions,
   - the maximum number of sessions active in the system,
   - the maximum number of sessions concurrently active in SCP,
   - the number of SCP system recoveries.

By design, session termination records are stored by session date and time. This means that parts 3 to 6 of a later session record always hold more current information than those of a previous one. Parts 3 to 6 of the record are used by the log file evaluation program to refresh the corresponding information provided; that is, information on the thread group, thread, roll facility and SCB.

This technique is used to keep up-to-date information about the Natural CICS system resources in case CICS terminates in an uncontrolled manner.

The session termination log records, of course, reflect only resources which have been used by the corresponding sessions. Therefore, these records may not reflect the full SCP environment. Reports of a full SCP environment can be obtained by making a snapshot of the whole environment by either using the *System Administration Facilities* function of the SYSTP utility (see the Natural *Utilities* documentation) or placing Natural under CICS into the CICS PLTSD (as described in the section ***Special Natural CICS Functionality***).

System snapshot records in the Natural log file represent session termination records without session-specific information as listed under part 2.

19 Natural CICS Performance Considerations

This chapter contains guidelines for setting up Natural in a CICS environment.

Choosing between the Roll Server and Roll Facilities

The Roll Server is versatile and scalable when used for saving and restoring Natural session data over a screen I/O. You must use the Natural Roll Server when running Natural in a CICSPlex or Parallel Sysplex environment, and you can use the Roll Server locally in a single CICS region. The Roll Server is able to handle a large number of concurrently executing Natural sessions. However, using roll facilities to handle a large number of concurrently executing Natural sessions may cause resource contention.

Natural installations in different CICS regions can share the same Roll Server, provided they use the same subsystem ID, because the Roll Server relates to a specific subsystem. If the terminal IDs in CICS regions with Natural installations that share the same Roll Server are not unique across these regions, use the `NTCICSP` parameter `UNITID=ON` to ensure that a unique session ID can be created or use separate Roll Servers for each CICS region with different subsystem IDs.

In a non-Parallel-Sysplex environment, the Roll Server can operate without a roll file, using only the in-storage Local Roll Buffer (LRB) that is contained in a z/OS memory object allocated above the bar. In that case, the LRB should have at least as many slots as the maximum number of concurrently executing Natural sessions. The slot size should be at least large enough to accommodate an average compressed Natural thread. If any “reserve” slots are available, threads that do not fit into a single LRB slot will occupy additional slots.

If Natural for zIIP is installed, the Roll Server schedules thread compression to run on a zIIP.

To avoid setting up and maintaining a Roll Server for Natural installations and CICS regions with a small number of concurrently executing Natural sessions, you can use one of the supported roll facilities described in this section.

For more information about the Natural Roll Server, see *Operations > Natural Roll Server Functionality*.

Choosing the Roll Facility

This section descrcibes the supported roll facilities. To specify a roll facility for a thread group, use the `PRIMERF` parameter of the `NCMTGD` macro.

Software AG does not recommend using a roll facility for Natural installations and CICS regions with a large number of concurrently executing Natural sessions. In such a case, you can use the Natural Roll Server.

This section covers the following topics:

Using CICS Main Temporary Storage

With CICS main temporary storage specified as the roll facility (`NCMTGD` parameter `PRIMERF=MAIN`), no VSAM I/O activity or communication to a temporary storage server is required on rolling. Therefore, using `PRIMERF=MAIN` is faster than `PRIMERF=AUX`. Due to the amount of used main storage, some tuning may be required.

Main temporary storage is allocated in 64-bit (above-the-bar) storage, controlled by the CICS `TSMAINLIMIT` system initialization parameter and the z/OS `MEMLIMIT` parameter. The thread data to be rolled out is split into chunks with a maximum size of 32 KB.

However, if you specify `MEMOBJR=ON` (default) with the `NTCICSP` macro in addition to `PRIMERF=MAIN`, the thread data to be rolled out is copied in a single operation into above-the-bar shared DSA (GSDSA) storage as a contiguous piece of storage.

Using CICS Auxiliary Temporary Storage

When Natural CICS Interface uses auxiliary temporary storage as the roll facility (`NCMTGD` parameter `PRIMERF=AUX`), the control interval size for the file being used must be at least 4 KB. If auxiliary temporary storage is not available, main temporary storage is used instead.

If you use VSAM roll files (`NCMTGD` parameter `PRIMERF=VSAM`) and the defined VSAM roll files become full or unusable during a CICS session, Natural CICS Interface uses auxiliary temporary storage.

You must increase the number of VSAM buffers defined by the CICS system initialization parameter `TS` to a reasonable value to reduce the number of physical I/O operations. Check he CICS statistics for bottlenecks in this area.

Control Interval for VSAM Roll Files and Auxiliary Temporary Storage

When using VSAM files or auxiliary temporary storage as the roll facility (`PRIMERF=VSAM` or `PRIMERF=AUX`), specify the largest possible control interval size of 32 KB to minimize the number of I/O operations and the CPU overhead necessary to perform the rolling. You may specify a control interval size of less than 32 KB for better exploitation of disk tracks or the usage of virtual storage for the VSAM buffers.

VSAM Roll Files versus CICS Temporary Storage

With the same CISIZE/record size, temporary storage causes less CPU overhead than VSAM roll files:

To write *n* records to temporary storage you have to issue *n* +1 CICS requests (that is, 1 for `DELETQ` and *n* for `PUTQ`) while you have to issue 2 *n* requests for VSAM roll files because of the VSAM transaction logic: *n* times (`READ` for `UPDATE` plus `REWRITE`).

For VSAM update requests, a physical I/O is always performed, whereas for temporary storage (`AUX`) records, buffering takes place, so that in many cases, records to be read are still found in the buffers.

However, CICS temporary storage may become a bottleneck when it is also being used by other applications.

VSAM roll files for Natural can overcome this situation (although at the expense of additional VSAM buffer space) especially when I/O contention can be avoided. VSAM roll files with optimum/maximum CISIZE/record size are particularly efficient when this record size cannot be specified for the CICS temporary storage file due to other requirements.

CICS temporary storage should be used whenever it can be dedicated to Natural. If CICS temporary storage is also used by other applications, you should evaluate whether the Natural performance is better when using VSAM roll files.

If Natural with CICS temporary storage does not perform worse, you should choose CICS temporary storage as roll facility and use the “saved” VSAM roll file buffer space for more TS buffers or for an additional thread.

Using VSAM RRDS Roll Files

The VSAM roll files should be considered for normal CICS VSAM file tuning, for example, `BUFNO` and `STRNO` parameters in the FCT. The CICS shutdown statistics should be checked for bottlenecks in this area.

As the roll files serve as a kind of page data set for Natural, everything which slows down the Natural rolling should be avoided, as there is journaling and logging; dynamic transaction backout (DTB) and forward recovery for roll files is useless and only causes overhead.

In MRO Environments

For performance reasons the VSAM roll files should be defined in the same CICS system in which the Natural applications are running; MRO function shipping should not be invoked. CICS local shared resources (LSR) can be used if there are enough buffers available.

Separate LSR Pool for Natural

The definition of a separate LSR pool for Natural roll files is recommended, with the number of strings (STRNO) greater than the number of threads. The number of buffers should also be greater than the number of threads. A greater number of buffers increase the look-aside hit ratio.

Shared Storage Threads versus GETMAINed Threads

This covers section the following sections

Storage Usage

Shared storage threads are pre-allocated during Natural CICS system initialization, which means that the storage covered by these threads is dedicated to the Natural CICS system, regardless of whether there are active sessions or not. On the other hand, GETMAINed threads only exist while the CICS task is active.

Controlling Storage Usage

With shared storage threads (TYPE=SHR), Natural under CICS always uses what has been pre-allocated during the initialization of Natural; therefore, the size of storage used by Natural threads is easily predictable. For GETMAINed threads (TYPE=GETM), however, the actual storage used depends on the number of Natural sessions that are currently active.

Although Natural itself has no mechanism for setting the maximum number of GETMAINed threads, this can be controlled by grouping the Natural transaction codes into a TRANCLASS. When a transaction code belongs to such a class, the maximum number of parallel tasks can be regulated by the MAXACTIVE parameter in the TRANCLASS definition.

Rolling

When a Natural session releases its shared storage thread, session data are kept in the thread in uncompressed format, unless another session needs to use this particular thread. If so, the new session is responsible for saving the old session's data.

Such an activity is called “deferred rolling”. It enables you to eliminate rolling entirely, provided that the number of available threads is greater or equal to the number of concurrently active Natural sessions.

Conversely, sessions that use GETMAINed threads always save their data prior to the FREEMAIN operation at CICS task termination, which leads to a roll overhead regardless of the number of concurrently active Natural sessions.

In environments with high volumes of Natural transactions, there is practically no difference between saving session data via the “immediate” or the “deferred” rolling method.

In busy Natural environments with a high ratio of Natural sessions to program storage threads, there is more roll-in/roll-out overhead, since these threads are shared by several Natural sessions. A potential problem in this situation is thread contention caused by Natural tasks with long-running Adabas requests; that is, with many Adabas calls.

To prevent such tasks from “locking” a thread for too long, they can be forced to release their thread by using Natural profile parameter DBROLL appropriately.

For GETMAINed threads, however, contention between two or more Natural sessions never occurs, since a TYPE=GETM thread belongs exclusively to the Natural session it was allocated for.

TYPE=GETM threads can thus be considered “single-use” resources that are never shared, whereas TYPE=SHR threads can be considered “multi-use” resources that may be shared.

Considerations for CICS/TS

The most important feature of CICS/TS in z/OS is transaction isolation, which means that a task's storage can be protected against other tasks.

To take advantage of this facility with Natural, TYPE=GETM threads should be used, since these threads are subject to transaction isolation, whereas “shared” TYPE=SHR threads are not. Also additional CICS overhead occurs for TYPE=SHR threads with CICS/TS.

While the thread selection algorithm for TYPE=GETM threads is trivial (when a Natural task is started, a thread is allocated via CICS GETMAIN), for TYPE=SHR threads, it is more complicated: the Natural threads environment is managed by the environment-dependent nucleus (queueing and balancing), whereas CICS does not know anything about Natural threads. In contrast to TYPE=GETM threads, where CICS would release the thread at the latest at the end of the task, with TYPE=SHR threads, Natural has to assign/release them to/from their sessions. In order to do so, Natural maintains a list of thread control blocks (TCBs).

Although Natural always keeps an exit active to be able to release session resources unknown to CICS (for example, TYPE=SHR threads) in the case of abnormal task termination, situations may occur where a Natural task terminates without its thread being marked as free in the associated TCB (for example, if an EXEC CICS ABEND CANCEL request has been issued in a non-Natural program called by Natural, or if Natural sessions have been flushed by any KILL transactions of a performance monitor).

To prevent problems with threads inadvertently left busy, Natural under CICS always checks in its thread selection algorithm whether the CICS task associated to a busy thread is still existing; if not, the thread is released.

With CICS versions prior to CICS/TS, this checking for active CICS tasks was done by control-block jumping, which means that Natural was checking for an active task by testing the consistency of the task's EISTG, TCA and TQE control blocks. With CICS/TS, because of transaction isolation, the storage of another task may not be accessible at all.

To accomplish this function in CICS/TS, the environment-dependent nucleus issues an EXEC CICS INQUIRE STORAGE TASK() request for any thread identified as busy in the thread selection routine. This means that there may have been some CICS requests before the task is finally ENQueued for thread resources. The same CICS command is also used for the serialization of Natural sessions (for example, deferred rolling of TYPE=SHR threads).

Conclusion

Both TYPE=SHR and TYPE=GETM threads have their advantages and disadvantages. However, with CICS/TS, TYPE=GETM threads are preferred, because of:

- the support of transaction isolation (z/OS only),
- more CICS-like tuning possibilities,
- worse performance of TYPE=SHR threads.

CICS Parameter Settings

CICS SIT parameters AMXT and CMXT should be used to control the number of concurrent Natural tasks.

The number specified should be greater than the number of threads. You should also consider to specify a separate transaction class with a suitable CMXT parameter for asynchronous Natural tasks and for Natural Advanced Facilities spool tasks so as to prevent logouts of “normal” Natural terminal tasks by too many of such “background” tasks occupying threads. Special thread groups can be defined for these transactions.

CICS dumps for Natural transactions should be suppressed, unless requested from Software AG personnel for debugging purposes. Natural itself generates dumps (via EXEC CICS DUMP) for non-

program check abends, and also for program checks if the Natural session parameter DU is set to ON. When no Natural dump is generated, a CICS dump is redundant and just causes overhead (particularly when creating a system/region dump, since the whole CICS system is halted until the snap dump is completed).

CICS trace is essential when analyzing problems, although it introduces system overhead. Also CICS performance monitoring tools and accounting packages cause system overhead of up to 30 percent and more. Some packages internally turn on the CICS trace and then intercept it. You should be aware of this potential system overhead. Also remember that Natural CICS Interface uses the CICS command level application programming interface: CICS command level requests produce much more trace entries (apart from other overhead) than CICS macro level requests.

Line Compression Systems

Natural itself optimizes its data streams by means of RA (repeat to address) and other techniques as screen imaging etc. If other line compression systems are installed, the Natural transactions should be excluded from being processed by these systems, as this would introduce overhead without achieving any benefit.

Pseudo-Conversational versus Conversational Transactions

When resuming a session, conversational Natural tasks are locked to their initial thread, which means that a conversational task has to wait for this thread if it is currently not available. Pseudo-conversational Natural tasks, however, are flexible to roll into any available thread.

In other words, the “classical” advantage of conversational tasks - less I/O for saving/restoring data over screen I/O operations - does not apply for Natural because of its thread technique.

Natural and Adabas

Since a Natural task in CICS waits for completion of an Adabas call, the servicing Adabas region/partition should always have higher priority than the CICS region/partition to minimize wait time.

CICS Monitoring Products

CICS monitoring products may offer facilities to purge CICS tasks, bypassing any abnormal termination exit set by the application.

Caution: Such facilities should not be used to cancel Natural tasks, as Natural may not be able to clean up its resources, and, even worse, the Natural CICS system may be left in an inconsistent state depending on what this task was doing.

To cancel Natural sessions, the Cancel/Flush Session functions of the SYSTP utility should be used instead; see the relevant section in the Natural *Utilities* documentation for details.

20 Natural Print and Work Files under CICS

This chapter discusses the use of Natural print and work files under CICS.

Customizing Print and Work File Usage

Natural CICS Interface supports Natural print and work files in CICS either as CICS transient data queues or as CICS temporary storage queues, both auxiliary and main.

To customize usage, set the following subparameters in the `PRINT` and `WORK` profile parameter:

```
AM=CICS, TYPE=TD/AUX/MAIN, DEST=queuename
```

For more information, follow the links shown below:

- `WORK` profile parameter description and how to set the above subparameter values, see the `NTWORK` parameter macro.
- `PRINT` profile parameter description and how to set the above subparameter values, see the `NTPRINT` parameter macro.

Natural CICS Interface print file support has been provided for tracing and logging purposes. It is not intended for dealing with reports. In particular, the keyword parameters for `DEFINE PRINTER` such as `PRTY`, `CLASS`, `COPIES`, etc., are not honored at all.

CICS Temporary Storage Print and Work Files

CICS temporary storage queues, both auxiliary and main, for CICS print and work files are RECFM=V files by design, available for input and output.

Although in Natural under CICS there is no exclusive control of a specific TS queue by a Natural session, you can automatically create session- or terminal-dependent printfiles or work files by specifying the string defined in the `NTCICSP` macro parameter `TERMVAR` (`&TID` is the default) in the subparameter `DEST` of profile parameter `PRINT` or in the subparameter `DEST` of the profile parameter `WORK`. When such a string is found within the eight-character `DEST` subparameter, it is replaced by the actual terminal ID.

In CICSplex Environment

When running in a CICSplex environment, Natural print and work files in CICS temporary storage must be defined as `TYPE=SHARED` or `TYPE=REMOTE` in a CICS TST.

NCI System Queues

In Natural under CICS, NCI system queues cannot be accessed. (NCI system queues are TS queues with a prefix defined in the `TSKEY` parameter of macro `NCMDIR`.)

CICS Transient Data Print and Work Files

A CICS transient data queue for a Natural CICS print and work file must be defined in the CICS DCT. For indirect destinations, the attributes of the *base* destinations are propagated. In particular, the attributes of an *extra-partition* destination, such as `RECFM` or `TYPEFLE`, determine the Natural work file attributes.

Intra-partition destinations have `RECFM=V` set by design and are available for both input and output.

CICS transient data print and work files are “shared files” in the sense that more than one session may issue I/O operations against such a file.

III Natural under Com-plete/SMARTS

21 Natural under Com-plete/SMARTS

This document describes the functionality of the Natural Com-plete/SMARTS Interface (product code NCF) and the operation and individual components of Natural in a Com-plete environment.

Note: SMARTS is an acronym for "Software AG Multi-Architecture Runtime System". It constitutes a runtime layer that allows POSIX-like applications to run on mainframe operating systems. Software AG products communicate with the operating system through the SMARTS layer.

Related Documents:

- Com-plete documentation set for details of the Com-plete product
- *Online Processing* in the Natural *System Architecture* documentation
- *Installing Natural Com-plete/SMARTS Interface on z/OS* in the *Installation for z/OS* documentation.
- *SYSTP Utility* in the Natural *Utilities* documentation
- *Natural under Com-plete/SMARTS User Abend Codes* in the Natural *Codes and Messages* documentation

Driver Parameters for the Natural Com-plete/SMARTS Interface

For information on the driver parameters that are available for the Natural Com-plete/SMARTS Interface, refer to the description of profile parameter `COMP` or parameter macro `NTCOMP` in the *Parameter Reference* documentation. `COMP` (or `NTCOMP`) comprises several keyword subparameters whose use is explained in the following sections.

Use of the Abend Exits

The `ABEXIT` exits can generally be deactivated by setting the keyword subparameter `SPIEA` to `OFF`.

The `ABEXIT` exit is called during Com-plete's EOJ handling for an abnormal program termination processing.

By default, an `OCX` abend is interpreted by the `ABEXIT` exit routine.

- When running with `DU=ON`, the Natural session is dumped and terminated properly, with error message `NAT9974`.
- When running with `DU=SNAP`, the Natural session is dumped as with `DU=ON`, but the session is not terminated.

Note: After the dump is taken, the Natural session continues as if running with `DU=OFF`.

- When running with DU=ABEND, the Natural session is dumped without terminating the session.
- When running with DU=FORCE, the ABEXIT exit routine is disabled, an immediate dump during Com-plete is produced.
- If LE370=ON is specified in the NTCOMP macro, or is specified dynamically, and the abend occurs while an LE program has control, user-written or language-specific condition handlers are ignored. The abend is handled by the ABEXIT exit routine, the Natural error message NAT0950 or NAT9967 occurs.

If DU=OFF, Natural responds with error message NAT0950, NAT0954, NAT0955 or NAT0956, and the entire abend PSW and the Registers 0 to 15 are contained in the IOCB at offset x'290'.

Storage Usage

At session initialization, the amount of space defined with the keyword subparameter NTHSIZE is allocated as thread GETMAIN above or below the 16 MB line, depending on the keyword subparameter THABOVE, for usage by Natural.

The Natural profile parameter WPSIZE determines the sizes of below and above work pools. By default, the size of the below subpool is set to 32 KB.

Therefore, you must catalog the Natural Com-plete front part with the Com-plete utility ULIB, RG size = 36 KB or larger.

The remaining areas within the Com-plete thread parts below and/or above (Com-plete ULIB RG=*specification* and/or THABOVESIZE=*specification*) are used by Com-plete for the following:

- user subroutines,
- increasing of variable buffers inside the Natural thread,
- subproducts doing "physical" GETMAIN requests, this enforces the Natural work pool allocation.

For more details concerning the ULIB RG and THABOVESIZE parameters, refer to the *Com-plete Utilities* documentation.

Support for Back-end Programs

Natural passes the following string to a back-end program:

- the Natural return code (fullword),
- the Natural termination message (A72),
- the length of the termination area (fullword),

- the termination data.

This string is mapped by the NAMBCKP macro.

The XNCFBACK source module is an example of a Natural back-end program in a Com-plete environment. It is written as reentrant code and can be loaded as RESIDENTPAGE program or once per user.

Com-plete Support in Natural Batch Runs

If you use the Com-plete services in a Natural batch run, the batch user ID remains logged on at the end of the batch run.

To avoid this situation, include the module COMPBTCH from the Com-plete distribution library in the batch Natural nucleus. This resolves the entry point for module EOJ, which is called at the end of the Natural batch job for termination clean-up.

The module NCFAM is used to access Com-plete print/work files. It has to be included in the linking of the Natural nucleus, together with the module COMPBTCH from the Com-plete distribution library.

Asynchronous Natural Processing under Com-plete/SMARTS

Asynchronous Natural processing is discussed in the section *Asynchronous Processing* in the Natural *Operations* documentation; however, some additional considerations apply when Natural is run under Com-plete.

Make sure that appropriate SENDER and OUTDEST destinations are specified for an asynchronous Natural session; otherwise, any output will lead to an abnormal termination.

In addition to Com-plete terminal IDs for SENDER and OUTDEST, the following keywords are supported by the Natural Com-plete/SMARTS interface:

Profile Parameter	Possible Values	Explanation
SENDER	DUMMY	Output is ignored.
	TID	Output destination.
	LUNAME	
OUTDEST	DUMMY	Output is ignored.
	CONSOLE	Output destination.
	LUNAME	

By default, the 3270 data stream protocol is used for output of an asynchronous Natural session running under Com-plete.

An example to start an asynchronous Natural transaction under Com-plete can be found in the library SYSEXTP, program ASYNCOMP.

Invoking Natural from User Programs

The Com-plete FETCH function is used to invoke Natural from a user front-end program under Com-plete; see the *Com-plete Application Programmer's Manual* for details.

Storage Thread Key Handling

If you want to use protection mode between Com-plete and your application program, you must set the Natural profile parameter SKEY to OFF in the Natural parameter module. The application program runs in the corresponding thread key.

You can improve performance of the application program under Com-plete by activating the Storage-Protection Override facility on your machine.

Set the thread key to 9 in the Com-plete startup parameter THREAD-GROUP for your Natural sub-group.

The front-end driver sets the Natural application automatically to the privileged mode if the thread key is 9, and uses the SPKA instruction for the key switch handling instead of the Com-plete function MODIFY with the function codes THRD/TCS.

Storage key switching is not performed for any Natural or editor buffer pool call.

Support for User Exit Handling during Session Initialization

During session initialization, it is possible to pass user-specific session information about the activation of a user exit to Natural. The exit is called before Natural has been initialized, after the driver/IOCB initialization is complete.

The driver passes as a parameter the address of the IOCB in Register 1, whereas the exit is activated/deactivated by the Com-plete functions COLOAD/CODEL; see the *Com-plete Application Programmer's Manual* for details.

The NCFUEXIT source module is an example of a user exit. The user exit module can be defined with the keyword subparameter EXIT.

Use of the SMARTS Server Environment

With the SMARTS Server Environment, it is possible to use the SMARTS portable file system as a container for input and output files as well as data sets on the native file system. It depends on the setting of the SMARTS parameters `CDI_DRIVER and MOUNT_FS` whether the environment variable refers to a the portable file system or to a native file system. For more information, see the *SMARTS Installation and Operations Manual*.

If environment variables are not defined, the normal data sets are accessed as described in the section *Data Sets Used by Natural under z/OS Batch* in the Natural *Operations* documentation.

The following topics are covered below:

Input/Output

Input/output in the SMARTS Server Environment is performed by DLL `NCF92IO`.

`NCF92IO` must be loaded into the resident area. If `NCF92IO` is loaded into the application program thread, the Natural session is terminated with a `NAT9980` error message.

Supported environment variables:

- **`CMPRINT` - Primary Report Output**
- **`CMSYNIN` - Primary Command Input**
- **`CMOBJIN` - Input for Natural `INPUT` Statements**

These environment variables are described below.

CMPRINT - Primary Report Output

Syntax:

```
CMPRINT=/pathname/filename[/],[mode]
```

Where

pathname	Specifies the location of the output file. If *pathname* refers to a portable file system, the path will be created; if it refers to a native data set, it must be available.
filename	Specifies the name of the output file. An asterisk (*) as the file name means that the file name is generated from the actual user ID. If *pathname* refers to the native file system and *filename* is terminated with the slash character (/), the sequential data set *pathname*/*filename* will be accessed; if it is not terminated with "/", the member *filename* in data set *pathname* will be accessed.
mode	Specifies the file mode as documented in the C Library for the `fopen()` function. The default is `w` (write).

Example: Assume `/fs/` is mapped to the native file system and `/pfs/` is mapped to a portable file system.

`CMPRINT=/fs/natural/test/print`	Member print in data set `natural.test` is accessed.
`CMPRINT=/fs/natural/test/print/`	Sequential data set `natural.test.print` is accessed.
`CMPRINT=/pfs/natural/test/print`	Member print in `/natural/test` of the portable file system is accessed.

CMSYNIN - Primary Command Input

Syntax:

```
CMSYNIN=/pathname/filename[/]
```

Specifies the *pathname* and *filename* of the appropriate command input file.

If pathname refers to the native file system and filename is terminated with the "/" character, the sequential data set *pathname*/*filename* will be accessed; if it is not terminated with a slash (/), the member *filename* in data set *pathname* will be accessed.

CMOBJIN - Input for Natural INPUT Statements

Syntax:

```
CMOBJIN=/pathname/filename[/]
```

Specifies the *pathname* and *filename* of the appropriate data input file.

If *pathname* refers to the native file system and *filename* is terminated with the slash character (/), the sequential data set *pathname*/*filename* will be accessed; if it is not terminated with a slash (/), the member *filename* in data set *pathname* will be accessed.

Print File/Work File

Print file and work file access in the SMARTS Server Environment is performed by DLL NCF92APS.

NCF92APS must be loaded into the resident area. If NCF92APS is loaded into the application program thread, the Natural session is terminated with a NAT9980 error message.

Supported environment variables:

- NAT_PRINT_ROOT - Path to the printer files on a PFS or native file system.
- NAT_WORK_ROOT - Path to the work files on a PFS or native file system.

Syntax Example:

```
NAT_WORK_ROOT=/qualifier/path1/path2
```

Where

qualifier	Determines whether a SMARTS portable file system or a native, OS-managed file system will be accessed.
path1/path2	Is the path to the location of the file in the appropriate file sytem.

Support for Com-plete's Recoverable Session Handling

To benefit from Com-plete's recoverable session handling available under z/OS, you have to link the module NCFROLLS to your front-end module. NCFROLLS serves as an interface to the *Natural Roll Server*, which has to be started to support recoverable sessions.

Furthermore, the module ATRRCSS needs not to be linked to your front-end module, because the RRS interface module is part of the Com-plete routine TLOPUSER. When a conversational terminal I/O is to be performed, the Natural thread is written to the Natural roll file in compressed form to allow resuming the Natural session after a Com-plete restart. For non-conversational terminal I/Os and thread locked applications, the Natural thread is not written to the Natural roll file; as a consequence, such sessions cannot be recovered.

Support for Natural for zIIP under Com-plete

Natural Com-plete/SMARTS Interface supports Natural for zIIP under Com-plete if the prerequisites described in the *Natural for zIIP* documentation are met.

Support of zIIP-enabled Natural Roll Server

Natural Com-plete/SMARTS Interface supports the Natural Roll Server. To benefit from this feature, Natural for zIIP under Com-plete and the Natural Roll Server NATRSM*vr* must be used. In this case the Compression and Decompression of the Natural thread is executed on IBM's System z Integrated Information Processors (zIIPs), thus reducing the CPU load of the GCPs.

To enable the use of the Natural Roll Server, just start the Natural Roll Server NATRSM*vr*.

Note: The notation *vr* represents the relevant product version (see also *Version* in the *Glossary*).

IV Natural under IMS TM

This document describes the functionality of the Natural IMS TM Interface (product code NII) and the operation and individual components of Natural in an IMS TM environment.

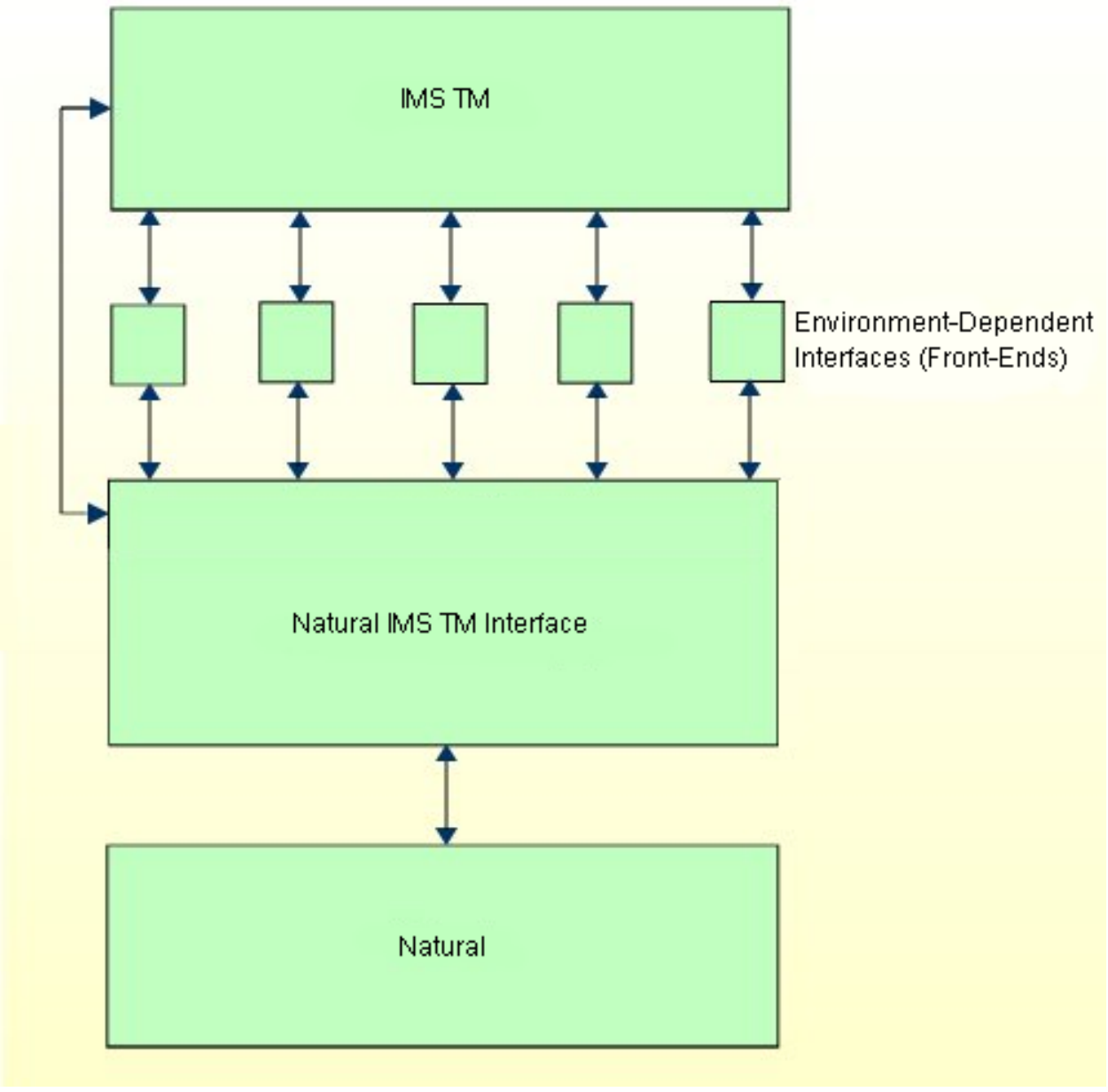
Related Documents:

- Installation - refer to *Installing Natural IMS TM Interface on z/OS* in the Natural *Installation for z/OS* documentation.
- Error Codes - for a list of the error codes and messages that may be issued by the Natural IMS TM Interface (NII), refer to *Natural under IMS TM Error Messages* in the Natural *Messages and Codes* documentation.

22 Natural under IMS TM - Environments

This chapter describes how Natural runs under various IMS TM environments.

IMS TM Interface Overview



IMS TM Environments

IMS TM provides three different types of environments:

- Natural in a Message Processing Region (MPP Environment)
- Natural in a Batch Message Processing Region (BMP Environment)

To be able to use Natural in each of these environments, different environment-specific interfaces are provided for the Natural IMS TM Interface. The task of such an interface is to receive input (usually a terminal input message) from the environment, to pass the input to Natural for processing and to direct the resulting output back to the correct destination (usually a terminal output message). This way, it is possible to use the functionality of Natural in all available IMS TM environments.

In addition to different available environments, within each environment, there are different ways of operating.

Natural in a Message Processing Region (MPP Environment)

In a message processing region, Natural online transactions can be one of the following:

- Dialog-Oriented Natural
- Message-Oriented Natural

Dialog-Oriented Natural

A dialog-oriented Natural session establishes an ongoing interaction with an IMS TM screen. Input and output messages to and from Natural are logically related and, across dialog steps, Natural saves information so as to be able to correctly process the next input message. In a dialog-oriented way, Natural can be run as either a conversational or a non-conversational transaction.

In a dialog-oriented environment, Natural can be executed in multiple-message processing regions, as Wait-for-Input (WFI) transaction and with the parallel-scheduling option.

To run Natural in dialog-oriented environments, you either have to use the roll server or roll files (see **The Roll File and Roll Server**).

If the Natural IMS TM Interface detects an error situation, a record containing information about this error situation is written to the IMS TM log file (see **Recovery Handling**). Thus, all terminals on which Natural is to be executed and all Natural transaction codes have to be authorized to issue the /LOG command using the Automated Operator Interface (AOI).

Message-Oriented Natural

A message-oriented Natural session processes non-3270-formatted messages from the IMS TM message queue. The input messages are considered to be unrelated to each other and are not part of a dialog. In a message-oriented way, Natural must be run as a non-conversational transaction.

Natural in a Batch Message Processing Region (BMP Environment)

In a batch message processing region, Natural can have access to the IMS TM message queue by using an input transaction code. With batch-oriented BMP regions, Natural supports symbolic checkpoint and extended restart. The input messages are non-3270-formatted messages.

Dialog-Oriented Environments

This section discusses special points valid for the dialog-oriented conversational environment only.

Special Considerations for a Conversational Environment

The dialog-oriented conversational environment is implemented by the Conversational MPP Interface which is linked with the Natural parameter module to the Conversational MPP Front-End. This front-end is the IMS TM application program and is scheduled by IMS TM if an input message for the assigned transaction code is available in the IMS TM message queue.

The dialog-oriented conversational environment requires a scratch pad area (SPA) of at least 157 bytes plus the `NRAST` value specified in the `NTIMSPT` macro of the Natural parameter module.

Special Considerations for a Non-Conversational Environment

The dialog-oriented non-conversational environment is implemented by the Non-Conversational MPP Interface which is linked with the Natural parameter module to the Non-Conversational MPP Front-End. This front-end is the IMS TM application program and is scheduled by IMS TM if an input message for the assigned transaction code is available in the IMS TM message queue.

When a dialog-oriented non-conversational environment is used, the *Natural Authorized Services Manager* with its SIP function enabled and the Physical Input Edit Routine are prerequisites.

- The *Natural Authorized Services Manager* is used to simulate the IMS TM SPA.
- The Physical Input Edit Routine is used to insert the transaction code in front of the input message.

You must specify the same Natural subsystem ID in the

- SPATID of the NTIMSPE macro (Natural parameter module),
- SPATID of the NIMPIXT macro,
- startup parameters of the Authorized Services Manager.

Special Considerations for an MSC Environment

Assuming the following environment, the Natural IMS TM Interface prepares the message X'000500006D' for NAT-B, which means that the terminal user has pressed CLEAR.

```
          IMS-A                          IMS-B
          -----                          -----
MPP-A1  MPP-A2  MPP-A3      MPP-B1  MPP-B2  MPP-B3
------  ------  ------      ------  ------  ------
                  DIRECT SWITCH
   NAT-A ----------------------------> NAT-B
```

Two entries must be created in the transaction code table: the first entry is for NAT-A, the second for NAT-B.

These two entries must specify different offsets for the Natural Reserved Area (NRA) and must ensure that these areas do not overlap.

NAT-B detects that a Natural session is to be started in IMS-B in the usual way and therefore gives control to its session-start exit routine. The session-start exit routine checks the input message for the string X'000500006D' and sets to 0 the length of the input message as seen by Natural.

If no additional logic is provided in either the exit NIIXSTAR or the exit NIIXSSTA, Natural starts a new user session in IMS-B.

It is assumed that IMS-A and IMS-B have different dedicated roll files allocated for Natural.

Both (or more) Natural sessions can communicate with each other by transferring data in the SPA when performing direct program-to-program switching.

For the time being, when two or more Natural sessions exist in such an environment, only the “active” session is terminated correctly.

Message-Oriented Environment

This section describes the message-oriented interface for use with Natural for IMS TM.

Introduction to the Message-Oriented Environment

This interface is designed to process nett-data input messages, which means that the messages do *not* represent a 3270 data stream. The message-oriented interface is driven by a user-written Natural program which instructs the interface to access the IMS TM message queue for the purpose of retrieving input messages.

The message-oriented interface has been created to support non-conversational, non-terminal driven transactions which must be executed as non-conversational MPP transactions.

Operation of the Message-Oriented Environment

The message-oriented interface incorporates functions from both the MPP and the BMP interfaces. The BMP interface is used as a basis, since much of the processing required emulates BMP-type transactions.

Since the message-oriented interface is not terminal-oriented, no messages or screen images are automatically generated to be sent to a terminal. The Natural nucleus is informed that it is running in a batch environment; therefore output is interpreted to be printer output and input is expected from a `CMSYNIN` file. All output which is normally written to `CMPRINT` is sent to the IMS TM destination specified with the Natural profile parameter `SENDER`. For details, see ***SENDER Destination*** below.

If Natural attempts to retrieve input data and no input data has been supplied by the application through the `STACK` command, EOF indicates that no input exists and Natural is terminated.

You can set `SENDER` to a new value at runtime by using the service module `CMSNFPRT`.

Except for checkpoint processing, Natural for Db2 processes transactions as if they were in BMP mode. This is necessary, since one physical scheduling can (and usually will) process several unrelated input messages. Under the conversational MPP interface, all transactions processed during one Natural session within this Natural session are considered to be related, requiring maintenance of database positioning and PCB usage.

Since transactions which are processed during one scheduling (and one Natural session) are not related to each other, the retention of Natural session information in the roll file is not required. Thus, no roll data set needs to be allocated for this interface. A roll slot area is still allocated via `GETMAIN` and used to store all Natural control blocks and work areas.

Since processing is performed on a message-by-message basis, there is no need for any relocation logic.

With the message-oriented interface, retrieval of all messages from the message queue is initiated by a front-end Natural program. This program must be user-written to meet your specific processing needs. However, it requires a specific structure, as shown in the following:

```
PROGRAM INITIALIZATION
REPEAT
CALL 'CMGETMSG' MESSAGE-AREA MESSAGE-LENGTH
IF MSG-LL = 0 /* QC on GU to message queue
TERMINATE
FETCH RETURN PGMA MESSAGE-AREA
REPEAT
CALL 'CMGETSEG' MESSAGE-AREA MESSAGE-LENGTH
IF MSG-LL = 0 /* QD on GN to message queue
ESCAPE
FETCH RETURN PGMB MESSAGE-AREA
END-REPEAT
END-REPEAT
END
```

The service module `CMGETMSG` reads the first message segment. The service module `CMGETSEG` reads all further message segments.

Since Natural cannot read input from `CMSYNIN`, it is required to use the Natural stack for input. This is done by using the Natural profile parameter `STACK`.

It is your responsibility to ensure that the IMS TM message queue is accessed by your application prior to the termination of Natural. If not, the Natural transaction abnormally ends with IMS TM abend code 462, indicating that a GU to the message queue has not been performed.

To obtain these Natural messages even in the case of an abnormal termination, you are recommended to define the first alternate PCB as an EXPRESS PCB.

The message-oriented environment is implemented by the NTRD Interface which is linked with the Natural parameter module to the NTRD Front-End. This front-end can either be called directly by IMS TM or via a bootstrap module that has been generated with the **NIMBOOT** macro.

If it is called directly by IMS TM, this front-end is the IMS TM application program which is scheduled by IMS TM if an input message for the assigned transaction code is available in the IMS TM message queue. You are recommended to use a Natural profile which contains the required `STACK` parameter. Specify `PROFILE=PROGRAM` in your Natural parameter module and create a profile with a name equal to the transaction code with which the interface is invoked. This way, you have the flexibility to use a different profile with a different `STACK` for each transaction code used.

If it is called via a bootstrap module, this bootstrap module is the IMS TM application program which is scheduled by IMS TM if an input message for the assigned transaction code is available

in the IMS TM message queue. This bootstrap module provides a string of dynamic profile parameters, one of which is the STACK profile parameter, and calls the NTRD front-end whose name is specified during the generation of the bootstrap module. If you want to call Natural with varying dynamic profile parameter settings, you must generate various bootstrap modules, each using its own string of dynamic profile parameters. Each of these bootstrap modules must be linked under a unique name. Also a unique IMS TM transaction code has to be assigned to each of the resultant load modules.

Batch Message Processing Environment

The Batch Message Processing (BMP) environment is implemented by the BMP Interface which is linked with the Natural parameter module and the work file/print file access routine NATWKFO to the BMP Front-End. This front-end is the IMS TM application program which is specified in the BMP JCL.

A standard batch Natural is executed in a Batch Message Processing region. In comparison with the standard batch Natural run, the optional input data set CONTROL may be used.

The optional BMP CONTROL file contains a maximum of two input cards.

- The first input card contains the following keyword parameters:

Keyword	Meaning
ENV-TAB=	The name of the environment table to be used.
TRNCODE=	The name of the transaction code to be used, see the TRNCODE parameter description.

Example:

```
ENV-TAB=ENVBMPO TRNCODE=NATIMS
```

- The second input card of the CONTROL file contains dynamic Natural parameters.

Using Both the CMPRMIN Data Set and the CONTROL File to Pass Dynamic Natural Parameters

If the CMPRMIN data set is also used to pass dynamic Natural parameters, the input of CONTROL is appended to the input of CMPRMIN. This means the parameters specified in CONTROL overwrite the parameters specified in CMPRMIN.

Working without CONTROL File

If the CONTROL file is not used, the name of the environment table is determined by the entry in the transaction code table which corresponds to the transaction code used (transaction-oriented BMP) or to the PSB name used (batch-oriented BMP).

Support of the Natural WRITE (n) Statement

With the WRITE (*n*) statement, up to 31 different reports on different printers can be produced within the same Natural program. The reports are sent to the IMS TM terminals specified either in the Natural parameter module or by using the Natural DEFINE PRINTER (*n*) statement. You have to specify AM=IMS in the NTPRINT macro which controls the report.

To be able to use this statement, define as many additional alternate TP-PCBs in your PSB as the number of parallel reports you want to create within the same Natural program, and specify the number of additional alternate TP-PCBs in your transaction code table by using the WRKPCBS keyword subparameter of the NTIMSPT macro (Natural parameter module).

Caution: Be aware that the first alternate TP-PCB is used by the Natural IMS TM Interface.

When using the WRITE (*n*) statement in a dialog-oriented environment, the following restriction applies:

The generation of a report cannot span one or more screen I/Os. If you want to use the same printer after a screen I/O, you have to close it explicitly before the screen I/O using the CLOSE PRINTER (*n*) statement.

To create reports, the following keyword subparameters of the NTPRINT macro are relevant:

Keyword Subparameter	Meaning
AM	Must be set to IMS.
DEST	Specifies the IMS TM destination.
BLKSIZE	Specifies the size of the buffer which is sent to the destination. Report lines are buffered.
DRIVER	Specifies the driver to be used to create the report. For a list of possible values, see the PRTDRIV keyword subparameter of the NTIMSPE macro in the Natural parameter module. The driver determines where you want to have the form feed (at the start of the report, the end, both the start and the end, or no form feed), where you want to start your page (on Line 1 or on Line 2 for compatibility with Natural IMS TM Interface Version 2.2) and where you want to print your report (SCS or non-SCS printer). In addition, you can specify that you want to use the JES API.
NAME FORMS DISP COPIES CLASS PRTY	These parameters are only evaluated if you use the JES API.

Hints Concerning NTPRINT and CLOSE PRINTER

NTPRINT Settings

You are strongly recommended that you always use the defaults for the OPEN and CLOSE subparameters in the NTPRINT macro or PRINT profile parameter definition for IMS TM printers (that is, for printers defined with AM=STD). This means either do not specify any values for OPEN and CLOSE or use te defaults OPEN=ACC and CLOSE=CMD.

This is especially important if you have statically defined a printer in the Natural parameter module for a different access method with other options for OPEN and CLOSE and if you dynamically overwrite the access method with AM=IMS. In this case, always specify AM=IMS,OPEN=ACC,CLOSE=CMD together.

Note: The NTPRINT options are merged with the dynamically specified PRINT options, even though the access method has been overwritten.

Problems which may occur with non-default values:

1. With OPEN=OBJ you may print to a wrong destination or get a NAT8211 error if the OUTPUT option has been specified in a DEFINE PRINTER statement. With OPEN=OBJ the printer is opened before the OUTPUT overwrite has been evaluated and the printer destination used is not the one which is specified in the OUTPUT option but the one specified with the PRINT parameter.
2. With CLOSE=FIN the printer is not closed at CLOSE PRINTER time but at FIN time. This means that the CLOSE may come after a GU has been issued to the message queue and the destination has been reset in the TP PCB. This will lead to NII error NII3641 for IMS TM status code QF (MPP) or A3 (BMP and OBMP/NTRD). With CLOSE=CMD, the printer is really closed with the CLOSE PRINTER statement.

Usage of CLOSE PRINTER or DEFINE PRINTER

A report written to an IMS TM printer is implicitely closed by IMS TM with the next GU call (that is, either at terminal I/O or through CMGETMSG). This means, IMS TM will print the report regardless of a CLOSE PRINTER or DEFINE PRINTER statement in the program.

For Natural, the printer is still open, and the next WRITE statement with the same report number will continue the already printed report which will lead to a NAT1518 error.

Scenario:

```
DEFINE PRINTER (1)
  WRITE (1) 'line 1'
  INPUT 'Press ENTER' or CALL 'CMGETMSG' (both issue a GU)
  WRITE (2) 'line 2'
```

The INPUT/CMGETMSG will "physically" close the printer and IMS TM will print a report containing the line 'line 1'.

As the printer is still "logically" open to Natural, the line 'line 2' will not start a new report and error NAT1518 will be caused as the destination is purged by the GU call.

You are therefore strongly recommended to observe the following rule:

Caution: A CLOSE PRINTER statement is required if, after the GU, the report with the same number is continued.

Please note that the DEFINE PRINTER statement does an implicit close in which case the CLOSE PRINTER statement is obsolete, for example:

Correct	Correct	Wrong (NAT1518)

REPEAT
  DEFINE PRINTER (1)
  WRITE (1)
 	
  INPUT
LOOP
	DEFINE PRINTER (1)
	REPEAT
	  WRITE(1)
	  CLOSE PRINTER (1)
	  INPUT
	LOOP
		DEFINE PRINTER (1)
		REPEAT
		  WRITE(1)
		
		  INPUT
		LOOP

SET CONTROL 'N' (Terminal Command %N)

The statement SET CONTROL 'N' (Terminal Command %N) does not apply under IMS TM. If it is used under IMS TM, it causes the next logical output screen to be suppressed.

Support of TS=ON for Natural under IMS TM Messages

All Natural under IMS TM messages are translated into upper case if TS=ON is specified in the Natural session.

SENDER Destination

In the message-oriented (NTRD) and in the server (SRVD) environment, all output that is normally written to CMPRINT is sent to the destination specified with the Natural profile parameter SENDER. With SENDER you either specify a valid IMS TM destination (usually an LTERM) to which the output is sent via the IMS TM message queue or you specify one of the following reserved values:

Value	Meaning
*WTO	Natural output is sent to the job log using a WTO with routing code 11 (programmer information).
*MTO	Natural output is sent to the IMS TM master console using a /BROADCAST MASTER command.
*PRINT*nn*	Natural output is written to Natural printer *nn*, that is the output is written using an internal WRITE(*nn*). The print file is closed after each output line and the carriage control character of each output line is the blank.

Notes:

1. If the /BROADCAST MASTER command fails (for example, due to authorization problems), an error message is issued using a WTO and all Natural messages, including the current message, are sent to the job log. That is, the SENDER destination is internally set to *WTO.
2. In case of *PRINT *nn*, you are strongly recommended to use only printers defined with AM=STD.
3. You are strongly recommended to code the SENDER destination in the Natural profile paramater module. This will ensure that the destination is found even if the Natural initialization fails (e.g due to an Adabas error NAT3048 or NAT3148).
4. If you really want to specify the SENDER destination dynamically, you must enclose the reserved values in single quotes (for example, SENDER='*WTO').

Support of Natural Profile Parameter PROGRAM

The Natural profile parameter PROGRAM is supported in the dialog-oriented environments and in the BMP environment.

In the BMP environment, the parameter PROGRAM behaves in the same way as in a standard z/OS batch environment. An example with name XNIIBACK is delivered in the NII*vrs*.SRCE data set. It is expected that the invoked back-end program returns to the Natural IMS TM Interface.

In a dialog-oriented environment, the Natural profile parameter PROGRAM behaves slightly different as compared to other Natural environments, including the BMP environment.

1. The name specified with the profile parameter `PROGRAM` or the Natural subprogram `CMPGMSET` is the name of an IMS TM transaction code.
2. The specified transaction code is only invoked if the Natural session terminates without an error.
3. The data supplied with the `TERMINATE` statement is passed as input message to the invoked IMS TM transaction.

Natural Development Server / Natural Web I/O Server Environment

This environment enables you to execute a Natural Development Server (NDV) or Natural Web I/O Interface (NWO) Server session under the control of IMS TM, and to have access to IMS TM ressources such as IMS TM DB or Db2 databases from within such a session. The server transaction is a non-conversational transaction.

For further information, see:

Introducing the Natural Web I/O Interface Server IMS Adapter , *Installing the Natural Web I/O Interface Server IMS Adapter under z/OS* and *Configuring the Natural Web I/O Interface Server IMS Adapter* in the *Natural Web I/O Interface* documentation.

Restriction for this environment:

Access to the IMS TM ressources is done using the user ID `EZAIMSLN`. This means no impersonation is used.

23 Natural under IMS TM - Components

This chapter describes the components of the Natural IMS TM Interface.

Front-End Module

The front-end module receives control from the IMS TM program controller DFSPPC20, except in the server environment where it is called by the call interface NIIBOOTS.

The front-end module must be built during the installation process of the Natural IMS TM Interface described in the *Installation for z/OS* documentation. The front-end-module consists of the following:

- Environment-Dependent Interfaces (Drivers)
- Natural Parameter Module
- Natural Work and Print File Access Method Module NATWKFO (AM=STD)

Environment-Dependent Interfaces (Drivers)

The environment-dependent interfaces are supplied as the following load modules:

- NIIBMP for the **batch message processing environment**,
- NIICONV for the **conversational dialog-oriented environment**,
- NIINONC for the **non-conversational dialog-oriented environment**,
- NIINTRD for the **message-oriented environment**,
- NIISFE for the **Natural Development Server/Natural Web I/O Interface server environment**, and
- NIISRVD for the **server environment**.

The load modules are supplied on the installation medium. You can configure the environment-dependent interfaces with the NTIMSP macro of the Natural parameter module. See also ***Natural under IMS TM - Configuration***.

Natural Parameter Module

The Natural parameter module is built during the installation. In addition to the parameter settings you need to adapt for your Natural environment, you must specify at least one NTIMSPT parameter macro for the Natural IMS TM Interface. See also ***Natural under IMS TM - Configuration***.

The Natural parameter module and the individual parameters and macros that can be specified with the Natural parameter module are described in the *Parameter Reference* documentation.

Natural Work and Print File Access Method Module NATWKFO (AM=STD)

The NATWKFO module is delivered as part of the base Natural. It is used for work file and print file handling for work files and print files defines with AM=STD. It is applicable to the BMP environment, including off-line IMS batch regions, the message-oriented environment and the server environment. It is not applicable to the dialog-oriented environments.

Some Natural products, such as Natural for Db2, require that their modules be linked to the Natural IMS TM front-end module. For further information, see the appropriate product documentation.

Natural IMS TM Interface Module NIIINTFM

The Natural IMS TM Interface module has to be created during the installation process and is common to all environments.

The interface module consists of the following components:

- Natural IMS TM Nucleus NIINUC
- Message Text Module NIIMSGT

The interface module is fully reentrant and can run above the 16 MB line. It is therefore eligible for the ECSA in order to have only one copy of the interface module for all IMS TM environments.

Natural IMS TM Nucleus NIINUC

The Natural IMS TM nucleus NIINUC is delivered as a load module and contains all the runtime routines required by the Natural IMS TM Interface.

Message Text Module NIIMSGT

The message text module NIIMSGT is part of the Natural IMS TM Interface module and is supplied both as a load and a source module. For each possible Natural IMS TM runtime error, it contains the corresponding message text. Each entry is generated by the macro NIMMSGT.

For a detailed description of the macro NIMMSGT, see *NIMMSGT Macro Parameters*.

Physical Input Edit Routine

The physical input edit routine is required only in a dialog-oriented, non-conversational environment. It is used to insert the transaction code preceding the message sent to the terminal. This is required as Natural runs in MFS bypass mode and the message sent to the terminal does not contain a transaction code.

The physical input edit routine is generated by using the `NIMPIXT` macro. For further information on the `NIMPIXT` macro, see ***NIMPIXT Macro Parameters***.

Once the physical input edit routine is generated, its name must be specified in the `TYPE` or `LINEGRP` macros of your IMS TM system definition. For all terminals on which the non-conversational environment is supposed to run, you must enable physical editing by using the `EDIT` parameter in the `TERMINAL` macro.

User Message Table DFSCMTU0

The delivered user message table `DFSCMTU0` is required only in a dialog-oriented, non-conversational environment. It contains the error messages for errors detected by the physical input edit routine.

The user message table `DFSCMTU0` must be integrated into the existing user message table of your IMS TM installation. In case of conflicts with already existing user message numbers of your IMS TM installation you may change the message numbers of the delivered `DFSCMTU0` by modifying the `EQUATES PIXTE` and `SIPSE` to create new message number ranges. The new start value of the message number range must be specified in the `NIMPIXT` macro.

Roll File and Roll Server

These components are used in dialog-oriented environments only.

Natural session-related information is held in the Natural thread. With each terminal output, the content of the Natural thread is saved either in a roll file or by using the roll server. The medium is defined by the `ROLLSRV` keyword subparameter in the `NTIMSPE` macro (Natural parameter module) described in the *Parameter Reference* documentation.

Using Roll Files

To use roll files, the ROLLSRV keyword subparameter is set to OFF.

A roll slot in the roll file is reserved for each Natural user at Natural session initialization time. The identifier of the slot is the IMS TM LTERM at which the Natural session is started. You must therefore ensure that all terminals that use the same set of roll files have different LTERM names. This is always the case if the roll files are used by a single IMS TM. The slot is freed when the Natural session terminates normally. In case of an abnormal termination, the roll slot remains allocated, but will be reused when the same user (identified by his LTERM) starts a new Natural session.

Roll files are accessed under the DD statements ROLLF1 - ROLLF5. The number of roll files used is defined by the ROLLFN keyword subparameter in the NTIMSPE macro.

If your Natural transaction code is scheduled in more than one MPP region or if you switch between transaction codes running in different MPP regions, you have to use the same roll files in all MPP regions.

If you reformat the roll file(s), make sure that no Natural transactions are active. If a transaction is scheduled after the roll file has been reinitialized, it cannot locate its roll slot on the roll file and abnormally terminates. To avoid this problem, it is recommended that you cold-start IMS TM after the roll file has been reformatted.

The roll files used by Natural under IMS TM have the same layout as the roll files used by the Roll Server and are formatted by the same utility program.

Note: The roll files used by Natural under IMS TM must not be shared with the Roll Server. If you use roll files for Natural under IMS TM and the Roll Server at the same time, you must assign an own set of roll files to the Roll Server.

Using the Roll Server

To use the roll server, the ROLLSRV keyword subparameter is set to ON.

Instead of using roll files which have to be allocated to each MPP region, you can use the Natural roll server. The roll server offers the following advantages:

- No DD statements in each MPP region.
- One central address space is responsible for access to the roll files.
- Support of main storage buffers to reduce disk I/Os to the roll files.

A slot in the roll server is reserved for each Natural user at Natural session initialization time. The identifier of the slot (roll server user ID) is the IMS TM LTERM at which the session is started, concatenated with the z/OS host ID and the IMS TM subsystem ID of the IMS TM dependent region in which the corresponding Natural transaction is scheduled. The slot is freed when the Natural

session terminates normally. In case of an abnormal termination, the slot remains allocated, but will be reused when the same user (identified by his LTERM) starts a new Natural session.

In a z/OS Parallel Sysplex environment you must use the roll server.

For further information on roll files and the roll server, see *Roll Server* in the Natural *Operations* documentation.

Authorized Services Manager

The *Natural Authorized Services Manager* is required in the following cases:

- In a dialog-oriented, non-conversational environment; see ***Special Considerations for a Non-Conversational Environment***).
- If Monitoring or Broadcasting is used; see ***Monitoring*** or ***Broadcasting***.
- If Accounting is used and the accounting information is written to SMF; see ***Accounting***.
- If buffer pool propagation is used; see the profile parameter BPPROP.

In the first two cases, the optional SIP function must be made available during startup of the Authorized Services Manager.

In a z/OS Parallel Sysplex environment, the SIP must be located in a Coupling Facility.

Shared Natural Nucleus

In an IMS TM environment, the Natural nucleus is always separated from the environment-dependent interface (driver). This means that you have to install the shared Natural nucleus. The same Natural nucleus can be shared by all Natural IMS TM environments.

Natural Buffer Pool

Since Natural under IMS TM is executable in more than one MPP region, it is recommended that the Natural buffer pool be a global buffer pool.

Although you can use a local buffer pool, this is not recommended in terminal-driven environments for performance reasons.

For further information, see *Natural Global Buffer Pool* in the Natural *Operations* documentation.

Adabas Interface

In order to access the Natural system file and Adabas user files, the Adabas interface is required.

By default, the appropriate Adabas interface is dynamically loaded at runtime.

- In terminal-driven dialog-oriented environments, the Adabas IMS TM Interface module `ADALNI` is used.
- In all other environments, the Adabas batch interface module `ADALNK` is used.

You can overwrite the name of the Adabas interface to be used by specifying the Natural profile parameter `ADANAME`.

Caution: You must not use the reentrant version of either of these interface modules.

Preload List

It is no longer required to use a preload list with the Natural IMS TM Interface, but for performance reasons it is recommended that you add the names of the following modules to the preload list for the Natural regions:

- the Natural IMS TM front-ends,
- the Natural IMS TM Interface module,
- the Natural shared nucleus,
- the Adabas interface.

24 Natural under IMS TM - Configuration

The main parameter required to configure the Natural IMS TM Interface are specified with the following profile parameter macros of the Natural parameter module:

- NTIMSP for general parameter settings,
- NTIMSPE for specification of environment-specific parameter sets, and
- NTIMSPT for transaction code definitions.

Important: You must at least specify the NTIMSPT macro and define individual parameters for your Natural IMS TM transactions (there are no default values).

In addition to the Natural parameter macros, you can use the optional macros described in the following section. They are provided by the Natural IMS TM Interface.

NIMMSGT Macro Parameters

The macro NIMMSGT generates each entry in the message text module NIIMSGT which is part of the Natural IMS TM Interface module. Each generated entry provides a message text for each possible Natural IMS TM error number.

The NIMMSGT macro is specified in one of the following two ways:

```
Nerror-number [*] NIMMSGT  message-text
```

In this case, Natural under IMS TM will display the message text as defined. The message text may be up to 72 characters long.

```
Xerror-number [*] NIMMSGT message-text
```

In this case, Natural under IMS TM will append an error-specific reason code to the current message text. The message text may be up to 64 characters long.

If the error number is followed by an asterisk (*), a snap dump will be generated when an error occurs. You may adapt the message text to your own requirements. You may also add or delete the DUMP option of a specific error number. You must not modify the error number and the characters N or R that precede the error number.

NIMPIXT Macro Parameters

The NIMPIXT macro generates the ***Physical Input Edit Routine***.

The parameters which can be specified with the macro NIMPIXT are listed in alphabetical order below:

PIXTE | SIPSE | SPATID | WTO | USER

Parameter	Possible Value	Description	Default	Comment
PIXTE	1 - 999	Specifies the start value for error numbers if errors are detected by the physical input edit routine.	400	This value is added to the return code set by the physical input edit routine. The result is the IMS TM error message number in the user message table DFSCMTUO.
SIPSE	1 - 999	Specifies the start value for error numbers if errors are detected by the *Authorized Services Manager*.	500	This value is added to the return code set by the Authorized Services Manager. The result is the IMS TM error message number in the user message table DFSCMTUO.
SPATID	*xxxx*	Specifies the Natural subsystem ID for the *Authorized Services Manager* which is used to save the SPA for the non-conversational driver. Any string up to 4 characters is possible.	None	The value of this parameter must be the same as the value specified for the SPATID keyword subparameter in the NTIMSPE macro (Natural parameter module).
WTO	YES NO	Specifies whether a WTO message is issued if the *Authorized Services Manager* fails.	NO	None.
USER	*xxxxxxxx* NO	Specifies whether a user-specific physical input edit routine is to be called if the NIMPIXT macro does not find the SPA. If a user-specific input edit routine is to be called, specify the name of the routine.	NO	None.

NIMBOOT Macro Parameters

The macro NIMBOOT generates the bootstrap module used by the message-oriented environment or the server call interface used by the server environment.

NIMBOOT includes the following parameters:

TYPE | DRIVERN | ENVTNAM | TRNCODE | DYNPARM | SERVERN

Parameter	Possible Values	Default	Comment
TYPE	SERVER	Empty	TYPE specifies the type of the interface module to be generated. With TYPE=SERVER, the server call interface NIIBOOTS is generated.
	Empty		If nothing is specified, the bootstrap module used by the message-oriented environment is generated.
DRIVERN	Any valid z/OS module name	None	This parameter specifies the name of the front-end module. If TYPE=SERVER is specified, the front-end module must have been generated for the server environment. If no TYPE is specified the front-end module must have been generated for the message-oriented environemt.
ENVTNAM	Any valid z/OS module name	None	This parameter is only used by the bootstrap module for the message-orineted environment (TYPE is empty). This parameter specifies the name of the environment table. This parameter is optional. If it is not specified, the environment table is determined by the entry in the transaction code table which corresponds to the transaction code used.
TRNCODE			This parameter is only used by the bootstrap module for the message-oriented environment (TYPE is empty). This parameter specifies the name of the transaction code which is internally used by the Natural IMS TM Interface. This parameter is optional and is only honored if TRNCODE=ON is specified in the NTIMSP macro (Natural parameter module). If it is not specified or if TRNCODE=ON is specified in the NTIMSP macro, the transaction code returned by the IMS TM INQY call is used. The transaction code is used to determine the entry in the transaction code table.
DYNPARM	Any character string of up to 80 characters.	None	This parameter is only used by the bootstrap module for the message-oriented environment (TYPE is empty). This parameter is used to define a valid string of up to 80 characters of Natural dynamic parameters.

Parameter	Possible Values	Default	Comment
SERVERN	Any valid z/OS module name	NIIBOOTS	This parameter is only used by the server call interface (TYPE=SERVER). This parameter specifies the name of the server environment. It is only relevant if you want to use several Natural servers in the same region. In this case, you must generate multiple server call interfaces and specify a unique name with SERVERN for each each of them. See Special Functions, *Server Environment*.

25 Natural under IMS TM - Service Programs

This chapter describes the service programs of the Natural IMS TM Interface.

Introduction to the Natural IMS TM Interface Service Programs

Purpose of Natural IMS TM Interface Service Programs

Service programs are Natural subprograms which provide Natural under IMS TM with additional functionality. You can call them from within a Natural program using a standard `CALLNAT` statement.

Location of Service Programs

The service programs are provided in the library `SYSEXTP` and you must copy them to the `SYSTEM` or steplib library. Sample Natural programs to invoke the service programs are also provided in the library `SYSEXTP`.

Common Return Codes

The last parameter in each service program is the return code whose format is (I4).

The following return code values are common for all service programs:

0	OK
-1	Non-supported function. This is an internal error, please contact Software AG support.

For specific return code values, refer to the individual service program descriptions below.

Error Handling

If an error occurs, either a Natural error message is issued or the session is terminated with a Natural IMS TM error message; see *Natural under IMS TM Error Messages* in the Natural *Messages and Codes* documentation.

Description of the Natural IMS TM Interface Service Programs

The following service programs are described below:

`NIIBRCST` | `NIICMD` | `NIIDEFT` | `NIIDEFTX` | `NIIDIRT` | `NIIDIRTX` | `NIIEMOD` | `NIIGCMD` | `NIIGMSG` | `NIIGSEG` | `NIIGSPA` | `NIIIMSIN` | `NIIISRTF` | `NIIISRTM` | `NIIPCBAD` | `NIIPCOM` | `NIIPMSG` | `NIIPSBAD` | `NIIPSPA` | `NIIPURG` | `NIIRETRM` | `NIISASD` | `NIIU3962`

NIIBRCST - Send Passed Message to Terminal

Sends the passed message to the specified terminal using the message output descriptor specified in the `MOD_name` parameter.

The following parameters are provided:

Parameter	Format/Length
`Terminal_name`	(A8)
`Message`	(A1/1:V)
`Message_length`	(I4)
`MOD_name`	(A8)
`Return_code`	(I4)

Specific Return Code Values: None.

Sample Program: `NIPGMSG`

NIICMD - Pass IMS TM Command to IMS TM

Passes the IMS TM command specified to IMS TM. If there is a reply, it is moved into the reply area provided. If the reply does not fit into the reply area, it is truncated and the return code is set to 4.

The following parameters are provided:

Parameter	Format/Length	Type
`Command`	(A1/1:V)	Input
`Command_length`	(I4)	Input
`Reply_area`	(A1/1:V)	Input/Output
`Reply_area_length`	(I4)	Input
`Reply_length`	(I4)	Output
`Status_code`	(A2)	Output
`Return_code`	(I4)	Output

Specific Return Code Values: 4 (reply truncated)

Sample Program: `NIPCMD`

NIIDEFT - Prepare Deferred Switch to Natural Transaction Code

Prepares a deferred switch to the specified Natural transaction code. With the next terminal I/O, the output is sent to the terminal and the next input from this terminal is processed by the transaction code specified in the parameter `Transaction_code`.

The following parameters are provided:

Parameter	Format/Length	Type
`Transaction_code`	(A8)	Input
`Return_code`	(I4)	Output

Specific Return Code Values: None.

Sample Program: `NIPDEFT`

NIIDEFTX - Prepare Deferred Switch to Non-Natural Transaction Code

Prepares a deferred switch to a non-Natural transaction code. With the next terminal I/O, the output is sent to the terminal using the given `MOD_name` and the next input from this terminal is processed by the transaction code specified in the parameter `Transaction code`.

If the suspend flag is set to `Y`, the Natural session will be suspended and can be resumed later. If the Natural session is resumed, it will first issue the last Natural screen.

If the suspend flag is set to `Y` you may not switch from a conversational Natural session to a non-conversational transaction code. If you try to do so, a Natural error message is issued.

The following parameters are provided:

Parameter	Format/Length	Type
`Transaction_code`	(A8)	Input
`Transaction_type`	(A4)	Input Possible values: `CONV` for conversational `NONC` for non conversational
`Suspend_flag`	(A1)	Input Possible values:

Parameter	Format/Length	Type
		Y - the Natural session will be suspended else the Natural session will be terminated
MOD_name	(A8)	Input
Message	(A1/1:V)	Input
Message_length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPDEFTX

NIIDIRT - Prepare Direct Switch to Natural Transaction Code

Prepares a direct switch to a specified Natural transaction code. On the next terminal write, the CHNG command to the specified transaction code is issued and the Natural screen is inserted using the alternate TP PCB.

If you switch from a conversational Natural session to a non-conversational one, the conversation is terminated and a dummy message using MOD_name NIIMODNC is inserted. This message unprotects the screen temporarily, and is thus overwritten by the first screen of the non-conversational Natural session.

Parameter	Format/Length	Type
Transaction_code	(A8)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPDIRT

NIIDIRTX - Prepare Direct Switch to Transaction Code

Prepares a direct switch to the specified transaction code. On the next terminal write, the CHNG call for the new transaction code is issued and the message and or the SPA are inserted using the alternate TP PCB. The transaction type defines the type of the new transaction code.

- If you switch from a conversational transaction code to a non-conversational one, the conversation is finished by issuing a dummy message using MOD_name NIIMODN, which unprotects the screen

temporarily, thus it will be overwritten by the screen issued from the non conversational transaction code.

- If the suspend flag is set to `Y`, the Natural session is suspended and may be resumed at a later time. When the Natural session is resumed, the last Natural screen is issued.
- If the suspend flag is set to `Y` you may not switch from a conversational Natural to a non conversational transaction code. If you try to do so, a Natural error message will be issued.
- If message length is set to zero, no message at all is inserted. This however is only possible if you switch to a conversational transaction code.

The following parameters are provided:

Parameter	Format/Length	Type
`Transaction_code`	(A8)	Input
`Transaction_type`	(A4)	Input Possible values: `CONV` for conversational transaction code `NONC` for non-conversational transaction code
`Suspend_flag`	(A1)	Input Possible values: `Y` - the Natural session will be suspended else the Natural session will be terminated
`Message`	(A1/1:V)	Input
`Message_length`	(I4)	Input
`Return_code`	(I4)	Output

Specific Return Code Values: None.

Sample Program: `NIPDIRTX`

NIIEMOD - Modify Setting of Module Output Descriptor

Modifies the current setting of the module output descriptor to be used in the insertion of the last message in a Natural session and sets it to the value specified in the parameter `MOD_name`.

The following parameters are provided:

Parameter	Format/Length	Type
MOD_name	(A8)	Input
Return_code	(I4)	Output

Sample Program: NIPEMOD

NIIGCMD - Retrieve Next Reply Segment of Previous IMS TM Command

Retrieves the next reply segment of a previously issued IMS TM command. The length of the reply is return in the parameter reply length. If the reply does not fit into the reply area, the reply is truncated and return code 4 is issued.

The following parameters are provided:

Parameter	Format/Length	Type
Reply_area	(A1/1:V)	Input/Output
Reply_area_length	(I4)	Input
Reply_length	(I4)	Output
Status_code	(A2)	Output
Return_code	(I4)	Output

Specific Return Code Values: 4 (reply truncated)

Sample Program: NIPCMD

NIIGMSG - Retrieve First Segment Next Message

Retrieves the first segment of the next message from the message queue by issuing a GU. The message area will contain the retrieved message including the leading LLZZ bytes. If there are no messages in the message queue, LLZZ is set to zero.

The following parameters are provided:

Parameter	Format/Length	Type
Message_area	(A1/1:V)	Output
Message_area_length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Programs: NIPGMSG, NIPGSEG

NIIGSEG - Retrieve Next Segment of Input Message

Retrieves the next segment of the input message by issuing a GN call. The message area will contain the retrieved message including the leading LLZZ bytes. If there are no more message segments in the current message, LLZZ is set to zero.

The following parameters are provided:

Parameter	Format/Length	Type
Message_area	(A1/1:V)	Output
Message_area_length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPGSEG

NIIGSPA - Retrieve Data from SPA Beginning

Retrieves data from the SPA beginning at the specified offset in the specified length.

The following parameters are provided:

Parameter	Format/Length	Type
Offset	(I4)	Input
Length	(I4)	Input
Area	(A1/1:V)	Input/Output
Return_code	(I4)	Output

Specific Return Code Values: 4

The retrieved data resides entirely or partially within the part of the SPA reserved for Natural.

Sample Program: NIPGSPA

NIIIMSIN - Retrieve IMS TM Environment Info

Retrieves the IMS TM environment information using the INQY ENVIRON call. If you specify a Reply_area_length smaller than 102, the reply will be truncated and you will receive return code X'0100' with reason code X'000C'.

The following parameters are provided:

Parameter	Format/Length	Type
Reply_area	(A1/1:V)	Output
Reply_area_length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: *nnxx*

nn: The first two bytes contain the AIB return code. *xx*: The second two bytes contain the AIB reason code. AIB denotes "Application Interface Block" and is used when calling IMS TM through the AIBTDLI interface.

Sample Program: NIPIMSIN

NIIISRTF - Create Multi-Segment Messages

Creates multi-segment messages. NIIISRTF performs the CHNG call for the specified destination and inserts the first message segment without performing a PURG call. Further message segments may be inserted using NIIISRTM. The message has to be terminated using NIIPURG. The LLZZ bytes are created by the service module.

The following parameters are provided:

Parameter	Format/Length	Type
Destination	(A8)	Input
Message	(A1/1:V)	Input
Message_length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPISRTM

NIIISRTM - Insert Message Segment into Message Queue

Inserts the next message segment into the message queue without performing a CHNG or a PURG call. The LLZZ bytes are created by the service module.

The following parameters are provided:

Parameter	Format/Length	Type
Message	(A1/1:V)	Input
Message_length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPISRTM

NIIPCBAD - Return PSB Name and PCB Address

Returns the currently scheduled PSB name and the address of the PCB identified by the logical name. If the logical PCB name is not defined in the transaction code table, a Natural error message is issued.

The following parameters are provided:

Parameter	Format/Length	Type
PSB_name	(A8)	Output
Logical_PCB_name	(A8)	Input
PCB_address	(B4)	Output
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPPCBAD

NIIPCOM - Move Data to Reply Area

Moves the data provided in the data area into the reply area specified in the NIIBOOTS call at the specified offset in the specified length. NIIPCOM may be called from the server environment only.

The following parameters are provided:

Parameter	Format/Length	Type
Offset	(I4)	Input
Data_area	(A1/1:V)	Input
Length	(I4)	Input
Return_code	(I4)	Output

Specific Return Code Values: 4 (calling environment, not server environment)

Sample Program: NIPPCOM

NIIPMSG - Send Message

Sends a message using a given MOD_name to the destination which is represented by the I/O PCB. The message is taken from the message area in the specified message area length. The message area must not contain the leading LLZZ bytes. In this way you can send MFS-formatted output messages back to the originator of the input message.

The following parameters are provided:

Parameter	Format/Length	Type
Message	(A1/1:V)	Input
Message_length	(I4)	Input
MOD_name	(A8)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPPMSG

NIIPSBAD - Return PSB Address

Returns the address of the PSB which is the address of the PCB address list.

The following parameters are provided:

Parameter	Format/Length	Type
PSB_address	(B4)	Output
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIPBOOTS

NIIPSPA - Replace Data in SPA

Replaces the data located in the SPA at the specified offset in the given length by the data provided in the data area.

The following parameters are provided:

Parameter	Format/Length	Type
Offset	(I4)	Input
Length	(I4)	Input
Data_area	(A1/1:V)	Input
Return_code	(I4)	Output

Specific Return Code Values: None.

An attempt to override the header of the SPA (first 14 bytes) and/or data residing in the Natural-reserved area is refused and a Natural error message is issued.

Sample Program: NIPPSPA

NIIPURG - Issue PURG Call

Issues a `PURG` call.

The following parameter is provided:

Parameter	Format/Length	Type
`Return_code`	(I4)	Output

Specific Return Code Values: None.

Sample Program: `NIPISRTM`

NIIRETRM - Move Data into Message Area

Moves data from the input message beginning at the specified offset in the specified length into the provided message area.

The offset is calculated from the `LLZZ` bytes.

The following parameters are provided:

Parameter	Format/Length	Type
`Offset`	(I4)	Input
`Length`	(I4)	Input
`Message_area`	(A1/1:V)	Input/Output
`Return_code`	(I4)	Output

Specific Return Code Values: None.

Sample Program: `NIPRETRM`

NIISASD - Modify SENDER and OUTDEST Settings

Modifies the current setting of the Natural profile parameters `SENDER` and `OUTDEST`.

The following parameters are provided:

Parameter	Format/Length	Type
`Sender`	(A8)	Input
`Outdest`	(A8)	Input
`Return_code`	(I4)	Output

Specific Return Code Values: None.

Sample Program: `NIPNTRD`

NIIU3962 - Terminate Session

Terminates the session with user abend code U3962 and produces a dump.

The following parameter is provided:

Parameter	Format/Length	Type
`Return_code`	(I4)	Output

Specific Return Code Values: None.

Sample Program: `NIPU3962`

26 Natural under IMS TM - Service Modules

This chapter describes the service modules of the Natural IMS TM Interface.

Purpose of Service Modules

Service modules perform IMS TM-specific functions. They can be called from within a Natural program using the standard Natural `CALL` interface. Sample programs are loaded by a Natural `INPL` into the library `SYSEXTP`.

Service Module Descriptions

This section contains a detailed description of all the service modules in alphabetical order. This includes a list of the parameters available and the name of the module-relevant sample program.

CMCMMND - Issue IMS TM Operator Commands

The module `CMCMMND` issues IMS TM operator commands and returns the reply segments to the Natural user program.

The following parameters are provided:

Name	Format/Length	Type	Comment
`Command`		Input	
`Command length`	(B4)	Input	
`Reply`		Output	
`Length of reply area`	(B4)	Input	

The operator command contained in the command area is issued to IMS TM with the indicated length.

If the user has set a non-zero reply length, any reply segments from IMS TM are moved into the reply area over the maximum available length. If the reply area is at least two bytes long, the first two bytes contain the IMS TM status code after the command call has been issued. The two rightmost bytes of the `REPLGTH` field contain the effective length of the total reply moved into the `REPLY` field.

If the reply from IMS TM has to be truncated, this is indicated by setting `X'80'` in the leftmost byte of the `REPLGTH` field.

Sample Program: `NIPSCMND`

CMDEFSW - Deferred Transaction Switch to Natural Transaction Code

The module CMDEFSW performs a deferred transaction switch to a Natural transaction code.

The following parameter is provided:

Name	Format/Length	Type	Comment
Trancode		Input	

With the next terminal I/O, the output is sent to the terminal and the next input from this terminal is processed by the transaction code passed as parameter message.

CMDEFSWX - Deferred Transaction Switch to Non-Natural Transaction Code

The module CMDEFSWX performs a deferred switch to a non-Natural transaction code.

The following parameters are provided:

Name	Format/Length	Type	Comment
Trancode		Input	
Message		Input	
Message length		Input	
MOD name		Input	

With the next terminal I/O, the given message with the given MOD name is inserted and the Natural session is terminated.

If the new transaction code is a Natural transaction code, the message and the MOD name passed as parameters are ignored and CMDEFSWX works as CMDEFSW.

Sample Programs: NIPSDEFX.

CMDIRNMX - Switch to Another Conversational Transaction w/o Message

The module CMDIRNMX has the same functionality as CMDIRSWX, except that no message is inserted to the alternate PCB. Thus, the only parameter you have to provide is Trancode.

The following parameter is provided:

Name	Format/Length	Type	Comment
Trancode		Input	

CMDIRNMX can also be used to perform a direct switch to another Natural transaction code, because in this case, the CLEAR key is given as input message to Natural by default.

If you want to switch to a non-Natural transaction code, it is strongly recommended to use the TERMINATE statement in conjunction with service module CMTRNSET instead.

```
CALL 'CMTRNSET' TRANCODE /* set transaction code */
TERMINATE                /* terminate Natural and call TRANCODE */
```

CMDIRNMZ - Switch to Another Conversational Transaction w/o Message

The module CMDIRNMZ has the same functionality as CMDIRSWZ, except that no message is inserted to the alternate PCB. Thus, the only parameter you have to provide is Trancode.

The following parameter is provided:

Name	Format/Length	Type	Comment
Trancode		Input	

CMDIRSWX - Switch to Another Conversational Transaction w. Message

The module CMDIRSWX performs a direct switch to another conversational transaction and specifies a message that is to be passed on to this new transaction.

The following parameters are provided:

Name	Format/Length	Type	Comment
Trancode		Input	
Message		Input	
Message length	(B4)	Input	

At the next terminal I/O, a change call is executed against the alternate PCB to set its destination to the value of the Trancode field. The SPA and the message are then inserted into the alternate PCB.

The new transaction code is checked if it is a Natural or a non-Natural transaction code.

In the case of a non-Natural transaction code, the Natural session is terminated.

In the case of a Natural transaction code, the CLEAR key is passed to Natural as input message, which means that Natural reacts as if the terminal user pressed the CLEAR key. The type of the new transaction code is automatically honored.

If you want to switch to a non-Natural transaction code, it is strongly recommended to use the TERMINATE statement in conjunction with service module CMTRNSET instead.

```
CALL 'CMTRNSET' TRANCODE /* set transaction code */
TERMINATE 0 MESSAGE      /* terminate Natural and call TRANCODE with MESSAGE */
```

The message MESSAGE is passed in the length of the Natural variable Message to the transaction code Trancode. The return code of the TERMINATE statement must be zero. Otherwise, the Natural session is terminated with termination error message NAT9987 and the transaction code switch does not take place.

Sample Program: NIPSDIRX

CMDIRSWZ - Switch to Another Conversational Transaction w. Message

The module CMDIRSWZ has the same functionality as CMDIRSWX.

The following parameters are provided:

Name	Format/Length	Type	Comment
Trancode		Input	
Message		Input	
Message length	(B4)	Input	

The difference compared to CMDIRSWX is that, in case of a switch to a non-Natural transaction code, the current Natural session is not terminated. This is done with the following intention:

- A given Natural session gives control to a non-Natural transaction code; the session is not terminated.
- The non-Natural transaction performs a terminal I/O and then switches back to the original Natural transaction, passing data into the SPA.
- The Natural transaction does not start a new session, but continues the old session were it has left it. The roll slot is obtained from the Roll Server, and control is given to Natural, so as to continue with an existing session.

The non-Natural transaction code must pass the message LLZZD, where LL=H'0005', ZZ=X'0000' and D=X'6D' are simulating to Natural that the CLEAR key has been pressed. By making the Natural program sensitive to the CLEAR key, it is able to recognize that the called non-Natural transaction has come back and it can retrieve the data prepared by the non-Natural transaction for use in subsequent processing.

CMDIRSWZ cannot be used if the transaction code to switch to is a Natural transaction code.

Sample Program: NIPSDIFS

CMDISPCB - Get PCB Content

The module CMDISPCB is used to obtain the contents of a PCB.

The following parameters are provided:

Name	Format/Length	Type	Comment
PCB number	(B4)	Input	
Receiving area		Output	
Area length	(B4)	Input	

After the call is executed, the receiving area contains the contents of the PCB with the requested number in the requested length. A check is made to verify that the requested PCB is within your current PCB list. The first PCB is PCB number 1, the second PCB is PCB number 2, etc. If you specify an invalid number, the field PCB number is set to X'FFFFFFFF' and no further information is passed to your application program.

Sample Program: NIPSPCBD

CMEMOD - Modify MOD Name Dynamically

The module CMEMOD allows the MOD name to be modified dynamically for a given LTERM at the normal end of a Natural session.

The following parameter is provided:

Name	Format/Length	Type	Comment
MOD name	(A8)	Input	

At a normal end of a session, the environment-dependent interface inserts the message X'00060000403F' into the IOPCB, using the MOD name whose value is contained in MOD name parameter. This is intended to present a meaningful screen (for example, a general menu) to the terminal user so that he can continue working at the terminal.

CMGETMSG - Read Next Message

The module CMGETMSG reads the next message from the message queue.

The following parameters are provided:

Name	Format/Length	Type	Comment
Message area		Output	
Message area length	(B4)	Input	

The length is checked to see if the received message fits into the message area. The message is moved including the LLZZ bytes into the message area. If there are no more messages, LL=0 is moved into the message area.

If the message does not fit into the message area, a corresponding error message is returned.

Sample Programs: NIPSGETM and NIPSOBMP.

CMGETSEG - Read Next Segment

The module CMGETSEG reads the next segment of the current message from the message queue.

The following parameters are provided:

Name	Format/Length	Type	Comment
Message area		Output	
Message area length	(B4)	Input	

The length is checked to see if the received message fits into the message area. The message segment is moved into the message area including the LLZZ bytes. If there are no more message segments, LL=0 is moved into the message area.

If the message does not fit into the message area, a corresponding error message is returned.

All read message segments are kept as a concatenated string in the internal input message buffer whose size is specified by the keyword subparameter MISIZE of the NTIMPSPE macro (see the *Parameter Reference* documentation). If you want to avoid an overflow of the internal input message buffer, use the CMGSEGO module instead of CMGETSEG.

Sample Program: NIPSOBMP

CMGETSPA - Transfer Data from SPA

The module CMGETSPA transfers the data from the SPA starting from the given offset in the requested length into the receiving area.

The following parameters are provided:

Name	Format/Length	Type	Comment
Offset	(B4)	Input	
Length	(B4)	Input	
Area	(B4)	Output	

Sample Programs: NIPSGSPA and NIPSPSPA

CMGSEGO - Read Next Segment

The module CMGSEGO reads the next segment of the current message from the message queue.

The following parameters are provided:

Name	Format/Length	Type	Comment
Message area		Output	
Message area length	(B4)	Input	

The length is checked to see if the received message fits into the message area. The message segment is moved into the message area including the LLZZ bytes. If there are no more message segments, LL=0 is moved into the message area.

If the message does not fit into the message area, a corresponding error message is returned.

Only the first and the current message segments are kept in the internal input buffer whose size is specified by the MISIZE keyword subparameter of the NTIMPSPE macro (see the *Parameter Reference* documentation). If you want to keep all message segments, use the CMGETSEG module instead of CMGSEGO.

Sample Program: NIPSOBMP

CMIMSID - Get MVS Subsystem ID

The module CMIMSID enables Natural programs to obtain the MVS subsystem ID of the IMS TM system in which they are currently scheduled.

The following parameter is provided:

Name	Format/Length	Type	Comment
IMSID	(A4)	Output	

After the call is executed, the field IMSID contains the MVS subsystem ID of the IMS TM system in which you are currently scheduled.

The module CMIMSID depends upon an internal IMS TM control block. Therefore, it is an IMS TM release-dependent function that will be updated whenever possible.

CMIMSINF - System Environment Info

The module CMIMSINF provides system environment information.

The following parameters are provided:

Name	Format/Length	Type	Comment
IMSID	(A4)	Output	The IMS TM ID.
SUFFIX	(A2)	Output	The preload suffix.
APPLGNAM	(A8)	Output	The application group name.
APPLNAM	(A8)	Output	The application name.
NRENT	(B4)	Output	The number of reentrant modules preloaded.
NNONR	(B4)	Output	The number of non-reentrant modules preloaded.

CMIMSINF is also an IMS TM release-dependent module.

Sample Program: NIPSINF

CMPCBADR - Return PCB Address

The module CMPCBADR returns the address of a PCB which is identified by a logical name. The PSB name is also returned to the Natural program.

The following parameters are provided:

Name	Format/Length	Type	Comment
PSB name	(A8)	Output	
PCB name	(A8)	Input	
PCB address	(B4)	Output	

After the call is executed, the field PCBADR contains the address of the PCB identified in the table module by the logical name PCBNAME in the table entry that corresponds to the currently scheduled transaction code. If the logical name does not exist for this transaction code, X'FFFFFFFF' is returned in the PCBADR field. In any case, the field PSBNAME contains the name of the currently scheduled PSB.

Sample Program: NIPSPCBA

CMPRNTR - Change Default Hardcopy Destination

The module CMPRNTR changes the default hardcopy destination set by the module NIIIMSHC to the value passed as parameter.

The following parameter is provided:

Name	Format/Length	Type	Comment
Destination	(A8)	Input	

The module CMPRNTR is provided for compatibility reasons only; use the Natural statement SET CONTROL*hdest-id* instead.

CMPUTMSG - Insert Output Message into IO-PCB

The module CMPUTMSG can be used to insert any given output message of a given length into the IO-PCB, using any given MFS MOD name. In this way, you can send MFS-formatted output messages back to the originator of the input message.

CMPUTMSG takes the number of bytes as indicated in the message length from the message area and inserts them with the specified MOD name in the message queue. There is no restriction upon the length of the message, except that it has to fit into the input message area of the environment-dependent interface.

The following parameters are provided:

Name	Format/Length	Type	Comment
Message area		Input	
Message length	(B4)	Input	
MOD name		Input	

If a non-blank status code is returned in the IO-PCB, Natural error message NAT8272 is issued which contains the status code as variable part.

CMPUTSPA - Move Data into SPA

The module CMPUTSPA moves the data with the given length at the specified offset into the SPA.

The following parameters are provided:

Name	Format/Length	Type	Comment
Offset	(B4)	Input	
Length	(B4)	Input	
Data		Input	

A check is done if the specified offset points into the Natural Reserved Area (NRA) within the SPA. If yes, return code 4 is returned.

Sample Program: NIPSPSPA

CMQTRAN - Content of Current Transaction Code Table Entry

The module CMQTRAN returns the contents of the current entry within the transaction code table.

The following parameters are provided:

Name	Format/Length	Type	Comment
Transaction code		Output	The transaction code under which you are running.
Offset	(B2)	Output	The offset of the NRA with the SPA.
Length	(B2)	Output	The length of the NRA.
Uoffset	(B2)	Output	Not used.
PSB name		Output	The name of the scheduled PSB.
Number of PCBs		Output	The number of PCBs whose addresses you can obtain using the module CMPCBADR.

The logical names by which you can refer to PCBs in the module CMPCBADR are not returned because of security considerations; you should be informed by your system about which logical names you are allowed to refer to.

Sample Program: NIPSQTRA

CMQUEUE - Insert Message into Alternate PCB

The module CMQUEUE inserts a message into the specified alternate PCB.

The following parameters are provided:

Name	Format/Length	Type	Comment
Destination		Input	
Message		Input	
Message length	(B4)	Input	
TP PCB number	(B4)	Input	Optional

This call causes an immediate change call to set the destination of the specified alternate PCB to the value contained in the field Destination, after which the message is inserted into the alternate PCB with the indicated Message length.

The transaction code is inserted after the LLZZ bytes with a length of 8.

After a PURGE call has been issued, control is returned to the next instruction in the Natural program.

The message can have any length up to the size of the input message area (usually 8000 minus 12 bytes).

The alternate PCB to be used is specified with the last optional parameter. If no TP PCB number is specified with the call, the alternate TP PCB specified with the ALTPCB keyword subparameter of the NTIMSPT macro (Natural parameter module) is used.

Sample Program: NIPSQLOA

CMQUEUEX - Complete Control over Message Content

The module CMQUEUEX provides you with complete control over the contents of a message that is to be queued in the IMS TM input queue.

The following parameters are provided:

Name	Format/Length	Type	Comment
Destination		Input	
Message		Input	
Message length	(B4)	Input	
TP PCB number	(B4)	Input	Optional

This call causes an immediate change call to set the destination of the specified alternate PCB to the value contained in the field Destaination, after which the message is inserted into the alternate PCB with the indicated Message length after the LLZZ bytes. The difference compared to CMQUEUE is that the transaction code is *not* inserted after the LLZZ bytes.

After a PURGE call has been issued, control is returned to the next instruction in the Natural program. The message can have any length up to the size of the input message area (usually 8000 minus 12 bytes).

The alternate PCB to be used is specified with the last optional parameter. If no TP PCB number is specified with the call, the alternate TP PCB specified with the ALTPCB keyword subparameter of the NTIMSPT macro (Natural parameter module) is used.

Sample Program: NIPSQUEX

CMSNFPRT - Set Logical Device Name

The module CMSNFPRT sets the logical name of the device to which the Natural messages during the online BMP run is sent.

The following parameter is provided:

Name	Format/Length	Type	Comment
Printer name		Input	

Before calling CMSNFPRT, use the Natural profile parameter SENDER to define the default output destination.

Sample Program: NIPSOBMP

CMSVC13D - Terminate Natural Session

The module CMSVC13D terminates the Natural session with user abend code U3962 and produces a dump.

Parameters: None

Sample Program: None.

CMTRNSET - Insert SPA via Alternate PCB

When the Natural session is terminated normally, the Natural IMS TM Interface performs a direct program-to-program switch to the specified transaction code and inserts the SPA via the alternate PCB.

The following parameter is provided:

Name	Format/Length	Type	Comment
Trancode		Input	

Sample Program: NIPSEOSS

NIIDDEFS - Deferred Switch to Foreign Transaction

The module NIIDDEFS is similar to module CMDEFSWX. If you use NIIDDEFS to perform a deferred switch to a foreign transaction, the current Natural session is suspended, as with module CMDIRSWZ. The suspended Natural session can be resumed at any time by sending back to Natural a message containing the CLEAR key.

The following parameters are provided:

Name	Format/Length	Type	Comment
Transaction code		Input	The transaction code to switch to.
Message		Input	The message to be sent to the foreign transaction code.
Message length	(B4)	Input	
MOD name	(A8)	Input	
Transaction type	(A4)	Input	An A4 variable containing the string CONV if the foreign transaction is conversational and the string NONC if the foreign transaction is non-conversational.

Return Codes:

0	OK
4	The message length is greater than the size of the message area defined in the environment table.
8	You tried to do a deferred switch with suspend from a conversational Natural to a non-conversational foreign transaction, something which cannot be done.
12	The fifth parameter is invalid; it contains neither `CONV` nor `NONC`.

Sample Program: `NIPSDEFS`

NIIDPURG - Send Multi-Segment Message

The module `NIIDPURG` does not have parameters. It issues a `PURGE` call using the same alternate PCB that has been used with the `NIIDQUMS` call and sends multi-segment messages that have been created using the module `NIIDQUMS`.

Return Codes: Either bytes two and three of the 4-byte return code contain the status code, or the return code has the value 0.

Sample Program: `NIPSQLMS`

NIIDQUMS - Create Multi-Segment Message

This module creates multi-segment messages. It has basically the same functionality as the module `CMQUEUE`, with the difference that `NIIDQUMS` does not issue a `PURGE` call.

The following parameters are provided:

Name	Format/Length	Type	Comment
`Destination`		Input	
`Message`		Input	
`Message length`	(B4)	Input	
`TP PCB number`	(B4)	Input	Optional

It is your responsibility to issue the `PURGE` call using the module `NIIDPURG`.

The alternate PCB to be used is specified with the last optional parameter. If no TP PCB number is specified with the call, the alternate TP PCB specified with the `ALTPCB` keyword subparameter of the `NTIMSPT` macro (Natural parameter module) is used.

Sample Program: `NIPSQLMS`

NIIDSETT - Get Foreign Transaction Code

In order to perform a correct transaction switch to a foreign transaction code, the type of the foreign transaction code must be known. To obtain this type, the special-purpose module `NIIDSETT` can be used. If `NIIDSETT` is not used, the foreign transaction code is assumed to be of the same type as the invoking Natural transaction code. If this is not the case, there will be unpredictable results or the session will terminate abnormally.

The following parameter is provided:

Name	Format/Length	Type	Comment
`Transaction type`	(A4)	Input	Possible values: `CONV` for conversational, `NONC` for non-conversational.

27 Natural under IMS TM - User Exits

This chapter contains an overview of the user exits that are available with the Natural IMS TM Interface. For each exit, a source module with the same name is provided. Each source module contains a description of the parameter list and of the register conventions.

NIIXACCT

The exit is called before an accounting record is written to the IMS TM log or to SMF. Thus, it makes it possible to modify the content of an accounting record. If `NIIXACCT` returns a non-zero register 15, the accounting record is not written at all.

NIIXSTAR

The exit is called with each transaction step after the SPA and the message have been retrieved and the Natural thread has been rolled in and decompressed. Within this exit, the Natural IOCB and the driver work area are accessible.

A value of 12 in register 15 upon return of `NIIXSSTA` forces the Natural IMS TM Interface to terminate the Natural session. Any other non-zero value in register 15 forces the interface to issue the Natural IMS TM Interface error 3517 with the reason code containing the value in register 15.

Note: This exit is not called when a new Natural session is started.

NIIXSSTA

The exit is called when a new Natural user session has been started and the SPA and the Natural IOCB have been initialized.Within this exit, the Natural IOCB and the driver work area are accessible.

A value of 12 in register 15 upon return of `NIIXSSTA` forces the Natural IMS TM Interface to terminate the Natural session. Any other non-zero value in register 15 forces the interface to issue the Natural IMS TM Interface error 3509 with the reason code containing the value in register 15.

NIIXISRM

The exit is called before the insertion of the message into the IOPCB.

NIIXISRT

The exit is called before the insertion of the SPA into the IOPCB, even at the end of the Natural session. The end-of-session situation can be recognized by a blank transaction code.

NIIXTGU0

The exit is called when the service module `CMGETMSG` is used. `NIIXTGU0` receives control immediately after the GU call against the IOPCB, regardless of the status code.

NIIXJESA

The exit is called when the JES API is used for writing reports. It is called after the options string has been created and may be used to modify the options string.

NIIXPRT0

The exit is called when reports are directly written to IMS TM printers. It can be used to set the codes for “form feed” and “new line”.

NIIXRFNU

The exit is called when the new Natural session is assigned to a roll file. It can be used to calculate the number of the roll file to be used for this session.

NIIXTGN0

The exit is called when the service module `CMGSEGO` or `CMGETSEG` is used. `NIIXTGNO` receives control immediately after the message segment is retrieved, regardless of the status code.

28 Natural under IMS TM - Special Functions

This chapter describes the use of special functions available with the Natural IMS TM Interface.

Prerequisites

Some of these functions require the *Natural Authorized Services Manager* (ASM).

- If the ASM is required, it must have been started before the function is used.
- The Natural subsystem used by the ASM must be the same as the one used by the Natural session.
- For accounting and monitoring, the SIP server must have been enabled in addition.

Accounting

The accounting function is only available in dialog-oriented environments. It is activated by setting the environment table keyword subparameter `ACTACTV` to `ON` in the `NTIMSPE` macro of the Natural parameter module.

With each terminal I/O, information about the specific Natural session is written to the IMS TM log or to SMF, depending on the setting of the `ACTLOG` keyword subparameter in the `NTIMSPE` macro.

- If the `ACTACTV` keyword subparameter is set to `CMD`, a `/LOG` command is issued that writes the accounting record to the IMS TM log. All transaction codes must therefore be allowed to use the `/LOG` command. At the beginning of each record an 8-byte header is inserted. This header helps to easily select the accounting records using the IMS TM utility `DFSERA10`. The header string is defined by the environment table keyword subparameter `ACTAHDR` (`NTIMSPE` macro).
- If the `ACTACTV` keyword subparameter is set to `LOG`, the accounting record is written to the IMS TM log using the `LOG` call. With the keyword subparameter `ACTARID` (`NTIMSPE` macro), you specify the log code to be used.
- If the `ACTACTV` subparameter is set to `SMF`, the accounting record is written to SMF using the Authorized Services Manager. With the `ACTARID` subparameter, you specify the SMF record type to be used.

The following information about each Natural user session is stored with each terminal I/O:

- IMS TM ID of the IMS TM system in which the user is active,
- LTERM name of the IMS TM terminal on which the session was started,
- user ID of the user of the Natural session (taken from the IOPCB),
- number of dialog steps currently performed,
- currently active transaction code,

- currently active PSB name,
- current Natural library name to which the user is logged on,
- currently active Natural program name,
- non-Natural transaction code to which the session is possibly suspended to,
- time and date when the session was started,
- time and date of the last ENTER operation,
- DBID and FNR of the Natural system file (FNAT) for this session,
- DBID and FNR of the Natural user file (FUSER) for this session,
- DBID and FNR of the Natural dictionary file (FDIC) for this session,
- DBID and FNR of the Natural Security system file (FSEC) for this session,
- DBID and FNR of the Natural spool file (FSPOOL) for this session,
- DBID and FNR of the Super Natural system file for this session,
- last encountered Natural error number,
- compressed thread length of the last terminal output.

The information is mapped by the DSECT NIMACTR. There are two areas for storing the DBID and FNR of the Natural system files used. In the first area, one byte is used for each DBID and FNR; this is supported for compatibility reasons. In the second area, a fullword is used for each DBID and FNR to support Adabas Version 6 or higher. The accounting record is prefixed with a length and version field.

Before the accounting record is written to the IMS TM log, respectively to SMF, the user exit NIIXACCT is called. You can use this user exit to tailor the accounting record to your requirements. You may also append information to the accounting record. In this case, you must set the length field to the new length.

Since the accounting record is built in the command buffer, the total length must not exceed the value specified with keyword subparameter CMBSIZE (NTIMSPE macro of the Natural parameter module) minus 17 bytes. The maximum length allowed is passed as parameter.

If NIIXACCT returns with a non-zero value in register 15, no accounting record is written.

Monitoring

The monitoring function is only available in dialog-oriented environments. It is activated by setting the environment table keyword subparameter MONACTV to ON (NTIMSPE macro in the Natural parameter module) and uses the SIP function of the Authorized Services Manager. The Natural subsystem must be the same as the one used by the Natural session to be monitored.

You can follow the ongoing activity of all Natural sessions which use the same Natural subsystem by using the Monitoring (M) function of the SYSTP utility. For more information on this utility, see *SYSTP* in the Natural *Utilities* documentation. The SYSTP session must also use the same Natural subsystem.

Broadcasting

The broadcasting function is only available in dialog-oriented environments. It is activated by setting the environment table keyword subparameter BROACTV to ON (NTIMSPE macro in the Natural parameter module) and uses the SIP function of the Authorized Services Manager.

Once broadcasting is active, it is possible to send broadcast messages to targeted users of a given Natural subsystem. Such users can be:

- all users of the Natural subsystem to which the sender is connected;
- all users of the Natural subsystem within the same IMS TM system as the sender of the message;
- all users of the Natural subsystem within the same IMS TM system as the sender of the message, but additionally restricted to a given transaction code;
- all users of the Natural subsystem within the same IMS TM system as the sender of the message, but additionally restricted to a Natural application;
- all users of the Natural subsystem within the same IMS TM system as the sender of the message, but additionally restricted to a Natural application and to a given FUSER system file.

When a session comes to a terminal output, a check is made to see whether the session has to receive a message or not. If not, the normal Natural output is sent. If yes, the message is sent instead of the normal output and, when pressing ENTER, the Natural nucleus is instructed to re-send the last screen. In this way, you first see the message and afterwards receive the normal Natural output screen.

If more than one broadcast message is available, the messages are displayed one after the other until the last message has been shown. Afterwards, the normal Natural output screen is displayed.

A broadcast message will be displayed only if its expiration time specified in the message creation procedure has not been exceeded.

When a broadcast message is sent, you must press RESET before you can press ENTER again. All possible attention IDs have the same effect as pressing ENTER.

The utility SYSTP can be used to create broadcast messages and to display the contents of all active messages together with the LTERM/IMSID of the sender. The text of a message is limited to 72 bytes.

Messages to be broadcast are saved in a pool maintained by the SIP server. They remain there until you delete them using the SYSTP utility or until you shut down the Authorized Services Manager.

Caution: When a broadcast message is deleted or created, all expired messages are deleted as well.

Server Environment

The server environment allows 3GL applications to execute Natural programs using a call interface. It is available in all supported IMS TM environments and consists of the Natural IMS TM driver NIISRVD of the server call interface NIIBOOTS and of the service API NIIPCOM.

NIISRVD and NIIBOOTS are delivered as source modules and must be assembled and link-edited on your site. For details, see *Installing the Natural IMS TM Interface on z/OS* in the Natural *Installation for z/OS* documentation.

The server environment allows you to start a Natural session by calling NIIBOOTS from any 3GL program. After the Natural session has been started, it returns to the calling 3GL program and waits for further input. The input would normally be expected from CMSYNIN, which means that the 3GL program has to simulate Natural's primary input data set.

It is strongly recommended to always put the server Natural on the NEXT line. This allows the next call to NIIBOOTS to either execute a Natural command or a Natural program. Otherwise, the next call to NIIBOOTS would be treated as input for a Natural program which had been started by a previous call to NIIBOOTS.

Similarly as with the message-oriented interface, all output normally written to CMPRINT is sent to the IMS TM destination specified with the Natural profile parameter SENDER. For details about special destinations used by the Natural IMS TM Interface, refer to ***Sender Destination*** in the section *Natural under IMS TM - Environments*.

Caution: In an MPP Environment, the same server Natural will be used by all transactions scheduled in this region by default. If you want to use multiple server Naturals in the same MPP region, you must generate multiple server call interfaces. Each server call interface must be generated with a unique name specified with the NIMBOOT parameter SERVERN and

must be linked under a unique name. It is recommended to name the load module with the name specified with SERVERN.

Call Interface NIIBOOTS

NIIBOOTS is the default name as used in the documentation and in the delivered sample programs. This default name can be changed during installation.

NIIBOOTS requires the following parameters:

- the PSB address (the address of the PCB address list),
- the command area,
- the reply area.

In the command area, the following may be passed:

- the startup parameters,
- any Natural command followed by its input data,
- the NIIBOOTS-specific commands, such as STAT and REFR (in combination with the startup parameters).

The startup parameters are passed in two contiguous 80-byte areas. The first area contains the name of the environment table and the name of the transaction code to be used as follows:

```
ENV-TAB=environment-table-name
TRNCODE=transaction-code-name
```

The transaction code is only honored if TRNCODE=ON is specified in the NTIMSP macro in the Natural parameter module. For details about the usage of the transaction code, refer to the NIMBOOT macro in the section *Natural under IMS TM - Configuration*).

The second area contains the dynamic Natural parameters with which the Natural session is to be started.

The reply area is the area in which a reply is to be entered from the executed Natural program using the service API NIIPCOM.

Each time it is invoked, NIIBOOTS checks whether the server Natural has been initialized.

- If Natural has not been initialized, a new Natural session is started and the received command is passed to Natural as a dynamic parameter.
- If Natural has been initialized, the string received in the command area is passed to Natural as a Natural command or as a Natural program.

The NIIBOOTS-specific commands STAT and REFR do the following:

- STAT returns COLD in the reply area if Natural has not been initialized and WARM if it has been initialized.
- REFR forces the initialization/reinitialization of Natural, regardless of the current state of Natural.

ON ERROR Routine Recommended

It is highly recommended to use an ON ERROR routine in the executed Natural programs in order to give back to the calling 3GL program some information in the reply area using NIIPCOM.

Return Codes

NIIBOOTS passes the return code provided by Natural on the termination of Natural.

Sample Programs

To illustrate usage of NIIBOOTS and NIIPCOM, the sample programs NIPBOOTS and NIPPCOM are provided. NIPBOOTS plays the role of the calling 3GL program, NIPPCOM is a sample Natural program executed in the server environment and writes the string NIISRVR into the reply area. The ON ERROR routine places the Natural error number in the reply area.

With the sample programs, you can go through the following scenario:

1. Pass the command STAT. The string COLD is returned to the reply area.
2. Pass the command: STACK=(LOGON SYSEXTP),SENDER=SO201, where SO201 is the LTERM name of the assigned printer device in the server Natural. Natural will be initialized and will be ready to receive a Natural command in library SYSEXTP. The successful logon message is issued on the assigned printer. Nothing is returned in the reply area.
3. Pass the command STAT. The string WARM is returned to the reply area.
4. Pass the command NIPPCOM. Program NIPPCOM is executed and the string NIPSRVR is returned to the reply area. Natural is ready to accept the next command in library SYSEXTP.
5. Pass the command: REFR STACK=(LOGON SYSEXTP;NIPPCOM),SENDER=SO201

   Natural is reinitialized and program NIPPCOM in library SYSEXTP is executed. The reply area contains the string NIPSRVR.
6. Pass the command FIN. Natural is terminated and no information is passed to the reply area. The return code will contain the return code of the Natural termination. The Natural termination message is issued on the assigned printer device.
7. Pass the command STAT. The string COLD is returned to the reply area.

29 Natural under IMS TM - Recovery Handling

This chapter describes recovery handling in the Natural IMS TM Interface.

System and User Abends

The Natural IMS TM Interface is protected by an ESTAEX environment which takes control in case of an abend.

- If a user abend is detected, resources are cleaned up and the abend is percolated without giving control to Natural.
- If a system abend is detected, Natural is informed about the abend and, depending on the setting of the Natural profile parameter `DU`, Natural continues with an error message or terminates the session.
- If the support of IBM's Language Environement (LE) is enabled and the abend occurs while an LE program has control, user-written or language-specific condition handlers are honored and Natural is only informed about the abend if the condition is percolated by all LE condition handlers. In this case, the abend is handled by Natural in the following steps before the standard abend handling takes place:
  - the corresponding LE error message is written to `SYSOUT`,
  - an LE snap dump is written to `CEEDUMP` according to LE run-time option `TERMTHDACT`,
  - LE is instructed to resume processing after the Natural `CALL` statement,
  - a special Natural error message (NAT0950 if `DU=OFF` or NAT9967 if `DU=ON`) is issued which indicates the LE error number.

  In all cases, you can produce a dump which represents the situation at the time when the error occurred (register contents, PSW, etc.). The dump is produced if `DU=ON` or `DU=SNAP` or if the user abend has requested this.

Non-Recoverable Errors

A non-recoverable error is a logical error detected by the Natural IMS TM Interface which cannot be handled by Natural. These situations typically occur during startup, termination or terminal I/O. In all cases, the Natural runtime is not active and can thus not react to the error.

If a non-recoverable error is detected, the Natural IMS TM Interface issues an NII error and terminates the session. The error message is also written to the IMS TM log and to the system log. Depending on the dump option in the error message table, a snap dump is produced.

If you do not wish a message to be written to the IMS TM log, set the `ERRLHDR` keyword subparameter of the `NTIMSPE` macro (Natural parameter module) explicitely to null, that is, you specify `ERRLHDR=,`.

If it is not possible to send the error message (for example if the GU has failed), the session abends (user abend).

Recoverable Errors

If a logical error is detected by the Natural IMS TM Interface which can be handled by Natural, for example an invalid destination for a report, a Natural error message is issued and Natural proceeds with its standard error handling.

V Natural under TSO

30 Natural under TSO

This document describes the functionality of the Natural TSO Interface (product code NTI) and the operation and individual components of Natural under TSO in the operating system z/OS.

Related Documents:

- *Installing the Natural TSO Interface on z/OS* in the Natural *Installation for z/OS* documentation.

General Information about the Natural TSO Interface

The Natural TSO interface (load modules NATTSO and NATTSOL) consists of a number of service routines interfacing with the z/OS operating system, see also *Installing the Natural TSO Interface on z/OS* in the *Natural Installation for z/OS* documentation. The module NATTSOL contains functions for the IBM LE environment. You can either link NATTSO to the Natural nucleus or you can run it separately by connecting it with a shared Natural nucleus using profile parameter NUCNAME.

NATTSO is fully reentrant and can run above the 16 MB line. Multiple Natural sessions can be started in parallel within one TSO region, and you can toggle between the sessions by means of a SWAPKEY (see the corresponding subparameter of profile parameter TSOP in the *Parameter Reference* documentation).

Driver Parameters for the Natural TSO Interface

For information on the driver parameters that are available for the Natural TSO Interface, refer to the description of profile parameter TSOP or parameter macro NTTSOP in the *Parameter Reference* documentation.

Data Sets Used by Natural under TSO

The following data sets are required if certain functions are used during a Natural TSO session:

CMEDIT	Software AG Editor Work File
CMHCOPY	Hardcopy Print Output
CMPLOG	Dynamic Profile Parameter Report Output
CMPRMIN	Dynamic Profile Parameter Input
CMPRT*nn*	Additional Reports 01-31
CMTRACE	External Trace Output
NATRJE	Job Submit Output
STEPLIB	Load Library for External Modules

CMWKF*nn*	Work Files 01-32

These data sets are described below.

Unless otherwise stated below, the default DCB RECFM/LRECL information is as follows:

RECFM=FB and LRECL=80 for sequential input data sets

RECFM=FBA and LRECL=133 for sequential output data sets

CMEDIT - Software AG Editor Work File

The Software AG editor work file VSAM data set is required if a local or global Software AG editor buffer pool is to be used. If not defined in the JCL or by TSO command ALLOC, the name of the Editor work file specified by subparameter DSNAME of profile parameter EDBP or parameter macro NTEDBP is used by Natural to do the dynamic allocation for the Editor work file.

Alternatively, profile parameter EDPSIZE can be used to run with an auxiliary editor buffer pool, which does not require an editor work file. For more information about the installation of the Software AG editor, refer to *Installing the Software AG Editor on z/OS* in the Natural *Installation for z/OS* documentation.

CMHCOPY - Hardcopy Print Output

The default name of the hardcopy print output data set is CMHCOPY. It can be changed by one of the following:

- the DEST subparameter of profile parameter PRINT for Print File 0,
- the profile parameter HCDEST, which is an equivalent of PRINT=((0),DEST=...),
- the setting of the system variable *HARDCOPY during the session,
- the terminal command %H during the session.

The subparameters of the PRINT profile parameter for Print File 0 can be used to change the default values for the hardcopy data set. The default data set name CMHCOPY implies CLOSE=FIN for the hardcopy print data set, that is, after the data set is opened for output, any subsequent change of the hardcopy print output data set name is not honored. If a different name is defined at open time, the hardcopy data set will be closed upon the next terminal I/O.

During the session, the hardcopy data set can be released and reallocated (before open or after close) by the by dynamic allocation (see Natural Application Programming Interface USR2021N).

CMPLOG - Dynamic Profile Parameter Report Output

If the profile parameter PLOG is set to ON and data set CMPLOG is available, the evaluated dynamic profile parameters are written to this data set during session initialization. If data set CMPLOG is not available, the evaluated dynamic profile parameters are written to the TSO terminal in line mode.

CMPRMIN - Dynamic Profile Parameter Input

If available, this data set is read during session initialization to get dynamic profile parameters. Only the first 72 positions of each record are used to build a dynamic profile parameter string.

Any other profile parameters which are passed directly for the start of the Natural nucleus, for example, by the TSO CALL command, are concatenated at the end of the parameter string which is build from the input of CMPRMIN; that is, these can be used to overwrite the parameters from CMPRMIN.

CMPRTnn - Additional Reports 01-31

These data sets can be used by Natural print file statements like WRITE (*nn*). If no DCB information (for example, RECFM, LRECL, BLKSIZE) is available, the defaults are defined by the profile parameter PRINT or by the macro NTPRINT in the Natural parameter module. The print file data set names can be overwritten by subparameter DEST.

CMTRACE - External Trace Output

If the profile parameter ETRACE is set to ON, or if the equivalent terminal command %TRE+ was issued, any Natural trace output during the session is written to the CMTRACE data set. To define the Natural components that shall be traced, the profile parameter TRACE is required.

If data set CMTRACE is not available, it will be allocated dynamically as

```
//CMTRACE DD SYSOUT=*
```

when the first trace record is to be written.

NATRJE - Job Submit Output

This data set is used for the Natural job submitting utility. If it is not defined, it will be allocated dynamically as

```
//NATRJE DD SYSOUT=(A,INTRDR)
```

when the first job is submitted.

STEPLIB - Load Library for External Modules

STEPLIB is the default load library name for loading external modules, for example, the shared nucleus (profile parameter NUCNAME), a separate Adabas link routine module (profile parameter ADANAME), the session back-end program (profile parameter PROGRAM) and any external subprograms not linked to the Natural parameter module.

The load library name can be overwritten by profile parameter LIBNAM. The specified load library name must be defined in the TSO job control or by an ALLOC statement, for example, in the CLIST which starts the Natural session.

CMWKFnn - Work Files 01-32

These data sets can be used by Natural work file statements like READ WORK FILE *nn* and WRITE WORK *nn*. If no DCB information (RECFM, LRECL, BLKSIZE, etc.) is available in the JCL or in the VTOC entry for the data set, the defaults are defined by the WORK profile parameter or by the NTWORK macro in the Natural parameter module. The work file data set names can be overwritten by subparameter DEST.

Issuing TSO Commands from Natural under TSO

You can use the Natural example program TSO in library SYSEXTP to issue TSO commands; for example:

```
TSO LISTALC STATUS
```

If you enter TSO without parameters, a menu prompts you for a TSO command. To exit from the menu, enter a period (.) in the first position, or press PF3.