



natural

Version 4.1.2 for Mainframes | TP Monitor Interfaces

This document applies to Natural Version 4.1.2 for Mainframes and to all subsequent releases.

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

© Copyright Software AG 1979 - 2003.
All rights reserved.

The name Software AG and/or all Software AG product names are either trademarks or registered trademarks of Software AG. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

TP Monitor Interfaces - Overview	1
TP Monitor Interfaces - Overview	1
Using Natural with TP Monitors	2
Using Natural with TP Monitors	2
TP Monitor Systems Supported by Natural	2
Using Natural in a Teleprocessing Environment	2
Embedding Natural in a TP Environment	2
Calling Natural Transactions under a TP Monitor	2
Terminating a Natural Session	2
Example Programs	2
Natural under CICS - Overview	3
Natural under CICS - Overview	3
Natural CICS Interface Functionality	4
Natural CICS Interface Functionality	4
NCISTART - Natural CICS Interface	4
Natural Nucleus under CICS	4
System Control under CICS	5
OSCOR/GETVIS - Natural Components in CICS Dynamic or Operating System Storage	5
Natural Storage Threads under CICS	7
Natural Roll Facilities under CICS	8
CICS Roll Facilities	8
Natural Local Buffer Pool under CICS	9
Natural Swap Pool under CICS	9
NCITIDEX Terminal ID Exit Interface	9
NCIUIDEX - User ID Exit Interface	10
NATUEX1 - User Exit	10
Natural CICS Interface Debugging Facilities	10
Using the TPF Parameter	10
Using Asynchronous Natural Sessions	11
Natural CICS Generation Parameters	12
Natural CICS Generation Parameters	12
NCISCPCB Generation Parameters	12
NCMDIR Macro Parameters	13
CICSPLX - Switching of CICS Application Region	13
ROLLSRV - Roll Server Rolling	13
SIPSERV - Authorized Services Manager's Session Information Pool	14
TSKEY - Prefixes for Natural CICS Temporary Storage Key	14
TSRECSZ - Record Sizes for Main and Auxiliary Temporary Storage	15
USERS - Session Information Record	15
NCMTGD Macro Parameters	16
PFKEY - PF/PA Keys for Thread Group	16
PRIMERF - Natural CICS Primary Roll Facility	16
THRDSIZE - Thread Size	17
THREADS - Number of Threads or Tasks Per Thread Group	17
TRAN - Transaction IDs for Thread Group	18
TYPE - Thread Type for Group	18
XTRAN - Hexadecimal Transaction IDs for Thread Group	19
NTSWPRM Macro Parameters	19
NCIPAPM Generation Parameters	20
NCMPRM Macro Parameters	20
BACKEND - Back-End Program Invocation Control	20
BACKOUT - Backout Transaction in the Case of Unrecoverable Abends	21
CHAP - Change Task's Dispatching Priority	21

COMACAL - CICS COMMAREA Usage for Subroutine Calls	21
COMAMSG - CICS COMMAREA Usage for Termination Messages	21
COMARET - CICS COMMAREA Usage for Task Control	22
CONSOLE - CICS Terminal ID for the Operator Console	22
DUPTID - Handle Duplicate Terminal ID	22
FLDLN - Supply Field Length List On External Program Call	23
LOGDEST - Natural CICS Logging Destination	24
MSGDEST - Destination ID for Natural Error Message Logging	24
MSGPFX - Generate NCI Message Prefix for WTL Messages	24
MSGTRAN - Internal Message Switching Transaction ID	24
PREFIX - Common Prefix for Programs and Files	25
PSTRNID - Control of *INIT-PROGRAM Variable Setting	25
RESENDC - Check for Screen Re-sending after Subroutine Calls	25
RESENDS - Screen Re-send Check after Pseudo-Conversational Session Resume	25
RJEDEST - Name of the Natural CICS Submit Destination	26
RJEUSER - Submit to POWER User ID Setting	26
SIGNON - SIGNON Behavior	27
SLCALL - Standard Linkage Call	27
SNDLAST - LAST Option Usage for EXEC CICS SEND Commands	27
TERMVAR - Terminal ID Variable for Natural Work Files	28
TRANCHK - Check Input Map for Transaction ID	28
UCTRAN - Lower/Mixed Case Support in Natural	28
UNITID - Establish Unique Terminal IDs	29
NCISCPRI Warnings and Error Messages	30
NCISCPRI Warnings and Error Messages	30
NCISCPRI Warnings and Error Messages	30
Customizing VSAM RRDS Roll Files	32
Customizing VSAM RRDS Roll Files	32
Increasing the Number of VSAM RRDS Roll Files	32
Decreasing the Number of VSAM RRDS Roll Files	33
Changing the Characteristics of the VSAM RRDS Roll Files	33
Natural in CICS MRO Environments	34
Natural in CICS MRO Environments	34
NCIPARM Parameter COMARET Set to YES	34
NCIPARM Parameter COMARET Set to NO	34
CICS Node Error Program Considerations for Natural	36
CICS Node Error Program Considerations for Natural	36
Normal Situation	36
Situations Not under Control of Natural CICS Interface	36
Recovery Mechanisms	36
Special Considerations	37
Example Dummy Program	37
CICS 3270 Bridge Support	39
CICS 3270 Bridge Support	39
Default Support of CICS 3270 Bridge	39
Full CICS 3270 Bridge Support	39
NCIXFATU - NCI Source Module	39
Profile Parameter DSC=OFF Recommended	39
Special Natural CICS Functionality	40
Special Natural CICS Functionality	40
Calling Non-Natural Programs	40
Calling Non-Natural Programs via Standard Linkage Conventions	40
Calling Non-Natural Programs with Parameter Values in a COMMAREA	40
Dummy Screen I/Os with Natural under CICS	41
NCISTART - Natural CICS Nucleus	41

Natural CICS Sample Programs	43
Natural CICS Sample Programs	43
Sample Programs in Natural CICS Source Library	43
XNCI3GC1 - Subprogram Call	43
XNCI3GP1 - Subprogram Call	43
XNCIFRCL - Front-End Program	43
XNCIFRPL - Front-End Program	43
XNCIFRCS - Front-End Program	43
XNCIFRPS - Front-End Program	43
XNCIFRCX - Front-End Program	44
XNCIFRPX - Front-End Program	44
XNCIFRNL - Front-End Program	44
XNCIFRNS - Front-End Program	44
XNCIFRNP - Initialization Program	44
XNCIFRNX - Front-End Program	44
XNCIGNIT - "Good-Night" Program	44
XNCIBACK - Termination Data Dump	44
XNCIRDC1 - Exit for SYSRDC	45
XNCIUIDX - User Exit	45
XNCIUCTR - U/L Case Switch	45
XNCINEP1 - Node Error Program	45
XNCINEP2 - Node Error Program	45
XNCITIDX - User Exit	45
Sample Programs for Use with VSE/ESA	45
Invoking Natural from User Programs	46
Invoking Natural from User Programs	46
Commands for Activating a Natural Session	46
Using EXEC CICS XCTL or EXEC CICS LINK	46
Using EXEC CICS START	46
Sample Programs	46
Using the External Subroutine CMTASK	47
Front-End Parameters	47
Front-End Invoked via LINK	48
In CICSplex Environments	48
Front-End Invoked via START	48
Front-End Invoked via XCTL	48
Invoking Front-End Program as Back-End	48
Asynchronous Natural Processing under CICS	49
Asynchronous Natural Processing under CICS	49
Asynchronous Natural Processing	49
Asynchronous Natural Sessions under CICS	49
Testing and Debugging	50
Logging Natural Sessions under CICS	51
Logging Natural Sessions under CICS	51
Logging Facility	51
Natural Log File Definition	51
Natural Log Records	51
Natural CICS System Restart Record	52
Natural Session Termination Record	52
Natural CICS Performance Considerations	54
Natural CICS Performance Considerations	54
Environment-Specific Considerations	54
Choosing the Roll Facility	54
Control Interval	54
VSAM Roll Files versus CICS Temporary Storage	54
Using CICS Auxiliary Temporary Storage	55

Using CICS Main Temporary Storage	55
Using VSAM RRDS Roll Files	55
Using the Natural Swap Pool under CICS	56
Shared Storage Threads versus GETMAINEd Threads	56
Storage Usage	56
Controlling Storage Usage	56
Swapping/Rolling	57
Considerations for CICS 4.1 and above	57
Conclusion	58
CICS Parameter Settings	58
Line Compression Systems	58
Pseudo-Conversational versus Conversational Transactions	59
Natural and Adabas	59
CICS Monitoring Products	59
Natural Print and Work Files under CICS	60
Natural Print and Work Files under CICS	60
Customizing Print and Work File Usage	60
CICS Temporary Storage Print and Work Files	60
In CICSplex Environment	60
System Queues	60
CICS Transient Data Print and Work Files	61
Natural under Com-plete/SMARTS	62
Natural under Com-plete/SMARTS	62
Use of the Abend Exits	63
Storage Usage	63
Support of Back-end Programs	64
Com-plete Support in Natural Batch Runs	64
Support of Asynchronous Natural Processing	65
Invoking Natural from User Programs	65
Storage Thread Key Handling	65
Support of User Exit Handling during Session Initialization	66
Use of the SMARTS Server Environment	67
Input/Output	67
Print File/Work File	68
Natural under IMS/TM - Overview	70
Natural under IMS/TM - Overview	70
Natural under IMS/TM - Environments	71
Natural under IMS/TM - Environments	71
IMS/TM Interface Overview	71
IMS/TM Environments	72
Natural in a Message Processing Region (MPP Environment)	73
Natural in a Batch Message Processing Region (BMP Environment)	73
Natural in an Off-line DL/I Batch Region	73
Dialog-Oriented Environments	73
Special Considerations for a Conversational Environment	74
Special Considerations for a Non-Conversational Environment	74
Special Considerations for an MSC Environment	74
Message-Oriented Environment	75
Introduction to the Message-Oriented Environment	75
Operation of the Message-Oriented Environment	75
Batch Message Processing Environment	76
Support of the Natural WRITE (<i>n</i>) Statement	78
Hints Concerning NTPRINT and CLOSE PRINTER	78
SENDER Destination	80
Support of Natural Profile Parameter PROGRAM	80

Natural under IMS/TM - Components	82
Natural under IMS/TM - Components	82
Front-End Module	82
Environment-Dependent Interfaces (Drivers)	82
Natural Parameter Module NATPARM	82
Natural Work and Print File Access Method Module NATWKFO (AM=STD)	83
Natural IMS Interface Module NIIINTFM	83
Natural IMS Nucleus	83
Natural IMS Parameter Module NIIPARM	83
Transaction Code Table NIITRTAB	83
Message Text Module NIIMSGT	84
DL/I Language Interface ASMTDLI	84
Physical Input Edit Routine	84
Roll File and Roll Server	85
Using Roll Files	85
Using the Roll Server	85
Shared Natural Nucleus	86
Natural Buffer Pool	86
Adabas Interface	86
Preload List	86
Natural under IMS/TM - Configuration Macros	87
Natural under IMS/TM - Configuration Macros	87
NIMDRIV Macro Parameters	88
NIMPARM Macro Parameters	89
A	90
B - C	91
E - H	92
L - N	93
P	94
R - S	96
T - U	97
NIMTRNTG Macro Parameters	98
NIMLPCB Macro Parameters	101
NIMMSGT Macro Parameters	101
NIMPIXT Macro Parameters	102
NIMBOOT Macro Parameters	102
Natural under IMS/TM - Service Programs	104
Natural under IMS/TM - Service Programs	104
Introduction to the Natural IMS/TM Interface Service Programs	104
Purpose of Natural IMS/TM Interface Service Programs	104
Location of Service Programs	104
Common Return Codes	104
Error Handling	104
Description of the Natural IMS/TM Interface Service Programs	105
NIIBRCST - Send Passed Message to Terminal	105
NIICMD - Pass IMS Command to IMS	105
NIIDEFT - Prepare Deferred Switch to Natural Transaction Code	106
NIIDEFTX - Prepare Deferred Switch to Non-Natural Transaction Code	106
NIIDIRT - Prepare Direct Switch to Natural Transaction Code	107
NIIDIRTX - Prepare Direct Switch to Transaction Code	107
NIEMOD - Modify Setting of Module Output Descriptor	108
NIIGCMD - Retrieve Next Reply Segment of Previous IMS/TM Command	108
NIIGMSG - Retrieve First Segment Next Message	109
NIIGSEG - Retrieve Next Segment of Input Message	109
NIIGSPA - Retrieve Data from SPA Beginning	110
NIIIMSIN - Retrieve IMS Environment Info	110

NIISRTF - Create Multi-Segment Messages	110
NIISRTM - Insert Message Segment into Message Queue	111
NIIPCBAD - Return PSB Name and PCB Address	111
NIIPCOM - Move Data to Reply Area	112
NIIPMSG - Send Message	112
NIIPSBAD - Return PSB Address	112
NIIPSPA - Replace Data in SPA	114
NIIPURG - Issue PURG Call	114
NIIRETRM - Move Data into Message Area	114
NIISASD - Modify SENDER and OUTDEST Settings	115
NIU3962 - Terminate Session	115
Natural under IMS/TM - Service Modules	116
Natural under IMS/TM - Service Modules	116
Purpose of Service Modules	116
Service Module Descriptions	116
CMCMMND - Issue IMS Operator Commands	116
CMDEFSW - Deferred Transaction Switch to Natural Transaction Code	117
CMDEFSWX - Deferred Transaction Switch to Non-Natural Transaction Code	117
CMDIRNMX - Switch to Another Conversational Transaction w/o Message	118
CMDIRNMZ - Switch to Another Conversational Transaction w. Message	118
CMDIRSWX - Switch to Another Conversational Transaction w. Message	119
CMDIRSWZ - Switch to Another Conversational Transaction w. Message	120
CMDISPCB - Get PCB Content	121
CMEMOD - Modify MOD Name Dynamically	121
CMGETMSG - Read Next Message	122
CMGETSEG - Read Next Segment	122
CMGETSPA - Transfer Data from SPA	123
CMIMSID - Get MVS Subsystem ID	123
CMIMSINF - System Environment Info	124
CMPCBADR - Return PCB Address	124
CMPRNTR - Change Default Hardcopy Destination	125
CMPUTMSG - Insert Output Message into IO-PCB	125
CMPUTSPA - Move Data into SPA	126
CMQTRAN - Content of Current Transaction Code Table Entry	126
CMQUEUE - Insert Message into Alternate PCB	127
CMQUEUEX - Complete Control over Message Content	127
CMSNFPRT - Set Logical Device Name	129
CMSVC13D - Terminate Natural Session	129
CMTRNSET - Insert SPA via Alternate PCB	129
NIIDDEFS - Deferred Switch to Foreign Transaction	130
NIIDPURG - Send Multi-Segment Message	130
NIIDQUMS - Create Multi-Segment Message	131
NIIDSETT - Get Foreign Transaction Code	131
Natural under IMS/TM - User Exits	132
Natural under IMS/TM - User Exits	132
NIIXACCT	132
NIIXSTAR	132
NIIXMSSP	133
NIIXSSTA	133
NIIXISRM	133
NIIXISRT	133
NIIXTGU0	133
NIIXJESA	133
NIIXPRT0	133
NIIXRFNU	133
NIIXTGNO	133

Natural under IMS/TM - Special Functions	134
Natural under IMS/TM - Special Functions	134
Prerequisites	134
Accounting	134
Monitoring	135
Broadcasting	135
Multi-Session Feature	137
Functionality of the Multi-Session Feature	137
Session ID	138
Multi-Session Database	138
Server Environment	138
Call Interface NIIBOOTS	139
ON ERROR Routine Recommended	140
Return Codes	140
Natural under IMS/TM - Recovery Handling	141
Natural under IMS/TM - Recovery Handling	141
System and User Abends	142
Non-Recoverable Errors	142
Recoverable Errors	142
Natural under TIAM	143
Natural under TIAM	143
Structure of the Natural TIAM Interface	143
Parameters in Macro NAMTIAM	143
ADACOM	144
ADDBUFF	144
APPLNAM	145
ATTKEY	145
CLRKEY	145
CODE	145
CURPRO	146
DELETE	146
DYNPAR	146
HCASK	147
ILCS	147
LF	148
LINK	148
LINK2/LINK3/LINK4	148
NUCNAME	149
PARMOD	149
PFK	150
REFRKEY	150
REQMLOC	150
SYSDTA	151
TRACE	151
TTYLS	152
TTYPS	152
T975X	152
UMODE	152
USERID	153
Common Memory Pools under TIAM	153
Natural Shared Nucleus	153
Natural under TSO	154
Natural under TSO	154
General Information about the Natural TSO Interface	154
Natural TSO Interface Generation Parameters	154
ABEXIT - Abend Processing Mode	154

ALTSCRN - Session Screen Mode	154
FACOM - Use of FACOM Operating System	155
LBPNAME - Sharing of Local Buffer Pools	155
LE370 - Use of IBM Language Environment	155
NDBFSRV - Natural for DB2 File Server	155
SUBPOOL - Storage Subpool for GETMAIN Requests	156
SWAPKEY - TSO/ISPF Split Screen Feature Support	156
TTYTYPE - Default Device Type for TTY Terminals	156
Datasets Used by Natural under TSO	156
CMEDIT - Software AG Editor Work File	157
CMHCOPY - Hardcopy Print Output	157
CMPLOG - Dynamic Profile Parameter Report Output	157
CMPRMIN - Dynamic Profile Parameter Input	157
CMPRT nm - Additional Reports 01-31	158
CMTRACE - External Trace Output	158
NATRJE - Job Submit Output	158
STEPLIB - Load Library for External Modules	158
CMWKF nm - Work Files 01-32	158
Issuing TSO Commands from Natural under TSO	158
Natural under UTM - Overview	159
Natural under UTM - Overview	159
Natural under UTM - Part 1	161
Natural under UTM - Part 1	161
Structure of the Natural UTM Interface	162
Formatting Messages - FREXIT	163
Embedding Natural in a UTM Application	164
Common Memory Pools	165
Natural Buffer Pool under UTM	165
Natural Swap Pool under UTM	165
Loading Natural in a Common Memory Pool - Natural Load Pool	166
Natural Monitor Pool	166
Other Storage Areas	167
Natural User Thread	167
Natural User Work Area Asynchronous Write Buffer	167
Natural User Area for Asynchronous Transactions	167
Natural Roll File - LINK=PAMNAT	167
Generating KDCROOT	169
Defining the UTM Resources - KDCDEF	170
Special Definition for Type 9755/9756 Terminals	170
Treatment of K Keys and F Keys	170
Support of IBM Type 3270 Terminals	171
Support of TTY Terminals	172
UTM DC-Transaction Exit Routine NUERROR	173
UTM Startup Function	173
UTM Shutdown Function	174
Natural under UTM - Macro Keyword Parameters	177
Natural under UTM - Macro Keyword Parameters	177
NATUTM Macro Keyword Parameters	177
ADACALL - Access to Adabas	177
ADACOM - Adabas Link Module Usage	177
ADAPRI - Activation of Adabas Priority Control for UTM Application	178
ADAUTM - Synchronization of Async UTM/Adabas Transactions	178
APPLNAM - Name of Natural UTM Application	178
APRISTD - Adabas Priority for Standard UTM TAC	178
ASAPPLI - Name of Logical UTM Communications Partner	179
ASYNTAC - UTM Transaction Code for Asynchronous UTM Task or Application	179

BADTAC - Activation of UTM Function "BADTACS"	179
CDYNAM - Maximum Number of Programs to be Dynamically Loaded	179
CLRKEY - Activation/Deactivation of CLEAR Key	180
CURPRO - Cursor Positioning to Protected Field	180
ICONTRL - UTM Input Exit for Messages in Minus Format	180
INITPRG - Value for Natural Variable *INIT-PROGRAM	181
KB - Pass KB Address as First Parameter	181
KBSAVE - Saving of UTM KB via SPUT	181
KBUSEXT - Length of UTM KB User Extension	181
LFH - Use of Adabas LFH	182
LINK - Programs and Modules Called from Natural	182
LINK2/LINK3/LINK4 - Extensions of Keyword Parameter LINK	182
LOFFMAP - Format Name for Logoff Message	183
NATMON - Automatic Activation of Natural Monitor during Application Startup	183
NUAADDR - Natural User Thread Address	183
NUCNAME - Name of Bounded Reentrant Natural Module	184
PARMOD - Generation of Front-End and Reentrant Parts	184
PENDPR - Define UTM TAC for "PEND PR"	185
PFK - Function Key Modes	185
PRKEY - UTM Return Code for Function Key	185
REFRKEY - Definition of UTM Function Key	186
ROLLACC - Access Method for Natural Roll File	186
ROLLTSZ - Maximum Roll Thread Size	187
RSTCNT - Control of Restart Situations	187
RSTWARM - Control of Restart Situations	188
SCRNOPT - Terminal Types with Deactivated Natural Screen Optimization	188
SHUTALL - Name of User Exit	188
SHUTLST - Name of User Exit	189
SPOOL - Automatic Start and Termination of Printer Task	189
STRTALL - Name of User Exit for All UTM Tasks	190
STRTFST - Name of User Exit for First UTM Task	190
SVDYPRM - Save Area Length for Dynamic Natural Parameters	190
SWAMODE - Switching from 31 to 24-Bit Address Mode	191
SWDPAGE - Pageability of Swap Pool Main Directory	191
SWPUSID - Swap Pool User Identification	191
SYAPPLI - Name of Logical UTM Communications Partner	191
SYNTAC - UTM TAC for Sending Messages from Async to Sync UTM Applications	192
SYSLST - SYSLIST File Generation for UTM Task	192
TACEND - Action at PEND	192
TCLA1 - UTM TACs for Async Transaction w. Priority Level 1	192
TCLA2, TCLA3, TCLA4 - UTM TACs for Async Transaction w. Priority Levels 2, 3, 4	193
TCLS1 - UTM TACs for Async Transaction w. Priority Level 1	193
TCLS2, TCLS3, TCLS4 - UTM TACs for Async Transaction w. Priority Levels 2, 3, 4	193
TERMTAB - Terminal Control Table for Natural Roll File Management	194
TID - Adabas User ID Construction Method	194
TRACE - Trace File Number and Trace Print Record Length	196
TTYLS - Physical Line Size for TTY Devices	196
TTYPS - Physical Page Size for TTY Devices	196
UMODE - Session Processing Mode	197
NATUTM Macro Entries	198
CMKBADR - Current Address of UTM KB	198
User Area in the Swap Pool Directory	198
NURENT Macro Keyword Parameters	199
ACCNT - Call Logic for User Account Routine	199
ATTKEY - Attention Interrupt Key	199
AUTOLINK - Use of AUTOLINK Function	199

CALLM31 - Switching from 31 to 24-Bit Address Mode	200
CLR3270=xxx - Definition of CLEAR Key	200
EXTAPPL - UTM TERMIN Name of External DCAM or PDN Applications	200
FPUT - Operation Supplement for Printing via KDCS-Call FPUT	200
ILCS - Support of CRTE or ILCS	200
K2 - UTM Return Code for K2 Key	201
PARMOD - Generation of Front-End and Reentrant Parts	201
SCRNTRC - Tracing of Screen I/Os	201
SPOOL - Automatic Start and Termination of Printer Task	202
UINPEX - Name of User Exit	202
UOUTEX - Name of User Exit	202
Natural under UTM - Part 3	203
Natural under UTM - Part 3	203
User Exits	204
ACCEXIT - Macro NATUTM	204
ACCINIT - Macro NATUTM	204
INPTEX - Program FREXIT	204
RP2PRNT - Macro NURENT	204
RMSPOOL - Macros NATUTM and NURENT	205
SHUTALL - Macro NATUTM	206
SHUTLST - Macro NATUTM	206
STRTALL - Macro NATUTM	206
STRTFST - Macro NATUTM	206
TRMIOEX - Program FREXIT	206
UINPEX - Macro NURENT	207
UOUTEX - Macro NURENT	207
UVGEXIT - Macro NATUTM	207
WHCEXT - Macro NURENT	207
Asynchronous Transaction Processing under UTM	208
Asynchronous Processing within a Natural UTM Application	209
Asynchronous Processing between two Natural UTM Applications	211
Printing under UTM	217
Using Local Non-Spooled Printers	217
Using NATSPOOL (Natural Advanced Facilities)	218
Other Spooling Systems	218
Calling Non-Natural Programs	219
Calling UTM Chained Partial Programs	220
Calling Adabas from Non-Natural Programs in a Natural UTM Application	220
Terminating a UTM Task Abnormally	220
Natural under UTM - Part 4	221
Natural under UTM - Part 4	221
Accounting for Natural/UTM Applications	221
Structure of the Accounting Record	222
Utility Programs for Use with Natural/UTM	223
Utility Program NATDUE	223
Utility Program INPTEX	224
Utility Program NATPRNT	224
Utility Program UTMTAC	224
Utility Program TACSWTCH	225
Software Exchange	229
Program XAMDUSA	229
Program UTMCOB	229
Program UTMNAV	229
Program NUEXAMPL	230
Program ACCEXIT	230
Program TABMOD	230

UTM TACCLASS Concept - Priority Control	230
UTM TACCLASS Switch	240
Generating a Natural/UTM Application	242
Generating the Natural UTM Interface	243
Linking the Non-Reentrant Front-End Part and the Reentrant Part	244
Setting Up the Natural Roll File	244
The Start Job for a Natural UTM Application	244
Optimizing Natural UTM Applications	245
Several Applications with one Common Natural	246
Lists of Shared and Application-Specific Parameter Modules	247
Entering and Defining Dynamic Natural Parameters	247
UTM User Restart	247
Adabas Priority Control	248

TP Monitor Interfaces - Overview

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

It is organized in the following sections:

- Using Natural with TP Monitors provides general information on the usage of Natural with TP Monitors.
- Natural under CICS describes the functionality of the 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 TIAM describes how to run Natural under TIAM.
- Natural under TSO comprises general information about the Natural TSO Interface and Datasets.
- Natural under UTM describes how to run Natural under UTM.

For Natural under CMS, see Natural under VM/CMS (in the Natural Operations documentation for mainframes).

The TP Monitor Interfaces documentation is supplemented by the following documents:

- Natural Installation Guide for Mainframes
- Natural Operations for Mainframes
- Messages and Codes

Using Natural with TP Monitors

This section covers the following topics:

- TP Monitor Systems Supported by Natural
 - Using Natural in a Teleprocessing Environment
-

TP Monitor Systems Supported by Natural

Currently, Natural supports the following teleprocessing monitor systems:

CICS | CMS | Com-plete | IMS/TM | TIAM | TSO | UTM

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

See also: SYSTP Utility.

The Natural utility SYSTP provides various TP-monitor-specific functions. It is available under the TP monitors CICS, Com-plete, IMS/TM, TIAM, TSO and UTM.

Using Natural in a Teleprocessing Environment

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.

Calling Natural Transactions under a TP Monitor

The Natural transactions can be called by invoking the TP program called Natural and supplying the LOGON system command 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.

Terminating a Natural Session

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

Example Programs

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

Natural under CICS - Overview

The Natural CICS Interface documentation covers the following topics:

- Natural CICS Interface Functionality
- Natural CICS Generation Parameters
- NCISCPRI Warnings and Error Messages
- Customizing VSAM RRDS Roll Files
- Natural in CICS MRO Environments
- CICS Node Error Program Considerations for Natural
- CICS 3270 Bridge Support
- Special Natural CICS Functionality
- Natural CICS Sample Programs
- Invoking Natural from User Programs
- Asynchronous Natural Processing under CICS
- Logging Natural Sessions under CICS
- Natural CICS Performance Considerations
- Natural Print and Work Files Under CICS

References to CICS Tables

Where appropriate, any references to CICS tables (DCT, FCT, PCT, PPT, TCT, etc.) can be considered as references to the corresponding:

- assembly-type resource definitions,
- online resource definitions via CEDA,
- batch resource definitions via DFHCSDUP.

See also:

- Installing the Natural CICS Interface in the Natural Installation Guide for Mainframes.
- Natural under CICS Abend Codes and Error Messages (in the Messages and Codes documentation for mainframes)
- Error Messages from the Natural Swap Pool Manager Valid under CICS and UTM (in the Messages and Codes documentation for mainframes).
- SYSTP - this Natural utility provides various TP-monitor-specific functions.
- Natural as a Server under CICS

Natural CICS Interface Functionality

This part of the Natural CICS Interface documentation describes the functionality of the Natural CICS interface. It covers the following topics:

- NCISTART - Natural CICS Interface
- Natural Nucleus under CICS
- System Control under CICS
- OSCOR/GETVIS - Natural Components in CICS Dynamic or Operating System Storage
- Natural Storage Threads under CICS
- Natural Roll Facilities under CICS
- CICS Roll Facilities
- Natural Local Buffer Pool under CICS
- Natural Swap Pool under CICS
- NCITIDEX Terminal ID Exit Interface
- NCIUIDEX - User ID Exit Interface
- Natural CICS Interface Debugging Facilities

Related Documents

- **Installation** - refer to Installing the Natural CICS Interface in the Natural Installation Guide for Mainframes.
- **Utility** - refer to the Natural utility SYSTP which provides various TP-monitor-specific functions
- **Operation, Individual Components** - for information on operation and the individual components of Natural in a CICS environment, see also:
Node Error Program Considerations for Natural | CICS 3270 Bridge Considerations | 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 Work Files Under CICS

NCISTART - Natural CICS Interface

The Natural CICS Interface NCISTART is implemented in command level Assembler, thus allowing Natural to be compatible with the CICS Multiple Region Option and the debugging facility CEDF.

NCISTART 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 made from NCISTART.

Natural Nucleus under CICS

The Natural nucleus is a combination of the reentrant Natural module and various support routines, which are delivered as source programs requiring site-dependent assemblies and as load modules.

The CICS-related components of the Natural nucleus are:

- the Natural CICS Interface NCISTART,
- the Natural CICS parameter module NCIPARM,
- the NaturalCICS interface object-only part NCINUC.

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.

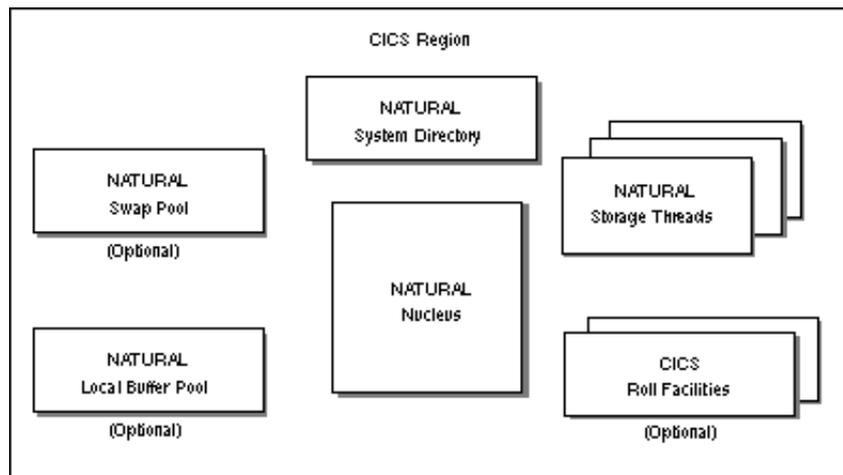
A complete understanding of system control can be attained from the following discussion of its structure and operation. Ensure that you understand this mechanism before starting the installation procedure of Natural under CICS.

OSCOR/GETVIS - Natural Components in CICS Dynamic or Operating System 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:

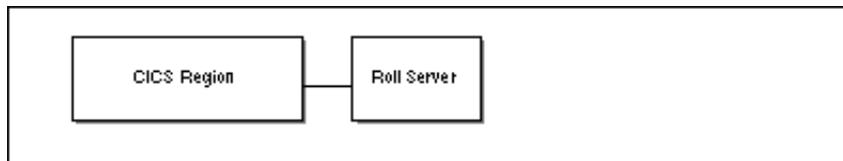
- Natural Storage Threads under CICS
- Natural Local Buffer Pool under CICS
- Natural Swap Pool under CICS
- Natural Roll Facilities



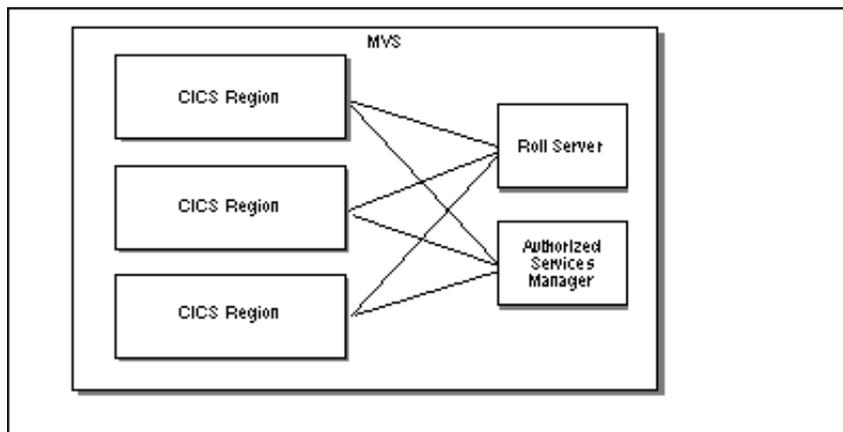
Scenario 1 applies when running Natural locally in a single CICS application region under OS/390 or VSE/ESA.

Platform:	Requirement:
OS/390 only	Additional scenarios are possible. The following three diagrams show combinations of OS/390 systems, CICS regions, the Natural Roll Server and the Natural Authorized Services Manager.

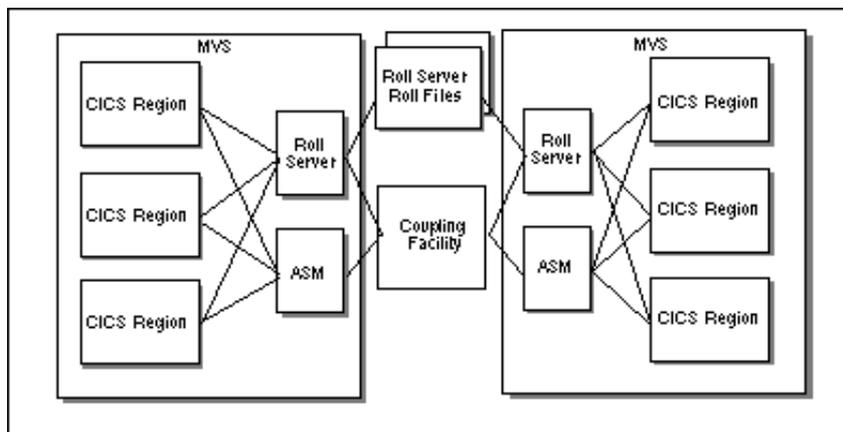
Scenario 2: Single OS/390 With Single CICS Region, Single Roll Server



Scenario 3: Single OS/390 With Multiple CICS Regions, Single Roll Server and (Optional) Authorized Services Manager



Scenario 4: Multiple OS/390 With Multiple CICS Regions, Multiple Roll Servers/Authorized Services Managers



In this scenario, you have to use the ADASVC Version 6.2.n and the Adabas link routine of the Adabas/CICS interface (ACI) Version 6.2.n.

Parameter Settings Required for the Above Scenarios

Module	Scenario 1	Scenario 2	Scenario 3	Scenario 4
(NT)BPI	TYPE=SWAP,SIZE= <i>nnn</i>	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	<i>name</i>
Authorized Services Manager/SIP	n/a	n/a	SIP slot number/size	XCF group name/ CF structure name

The 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 the 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).
- The 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/Os. 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 Natural utility SYSTP (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;
- the Natural swap pool.

See also the various component scenarios. For more information, see Roll Server (described in the Natural Operations for Mainframes 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 datasets (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.

When a swap pool is used, the CICS roll facilities only serve as backup for the swap pool. The choice of the roll medium is of greater importance when no swap pool is used, since it affects Natural performance and throughput.

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.



When using the Roll Server, the swap pool and 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, i.e., EXEC CICS GETMAIN SHARED with all CICS Transaction Server versions or a GETVIS request with CICS/VSE in VSE/ESA. 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 like batch Natural (OS/390 and VSE/ESA) or Natural/TSO or Natural/IMS (under OS/390 only).

Natural Swap Pool under CICS

The Natural swap pool offers the possibility to "swap" a compressed Natural session from the thread into a main storage area instead of doing expensive roll I/Os.

The swap pool is allocated as GETMAIN storage, i.e., EXEC CICS GETMAIN SHARED with all CICS Transaction Server versions or a GETVIS request with CICS/VSE in VSE/ESA. Sufficient storage must be available in the partition or in the relevant CICS DSA.

The options for the swap management are set in the Natural CICS source module "NCISCPCB" and by using the Natural profile parameter BPI.

The size, name and cache size of the swap pool are specified using profile parameter BPI or the corresponding macro NTBPI in the Natural parameter module NATPARM, that is, the (NT)BPI settings in effect for the Natural session initializing the NCI environment are taken.

For further details on the swap pool, see Natural Swap Pool (described in the Natural Operations for Mainframes documentation) and Using the Natural Swap Pool under CICS.

OS/390 systems:

The swap pool can only be used when running Natural under CICS locally in a single CICS region. However, even in such a scenario, you should consider using the Roll Server instead, because it runs asynchronously to the CICS region and because it can provide more roll buffers in its data space than the swap pool.

When using the Roll Server, the swap pool and the Roll Facilities are not available under CICS.

NCITIDEX Terminal ID Exit Interface

While the NCIUIDEX/NATUEX1 user exit interface can be used to set the internal terminal ID system variable *INIT-ID, the NCITIDEX terminal ID exit interface can be used to set the terminal ID which is then used by the Natural CICS interface NCI further on, i.e., this "logical" terminal ID will be used as part of the real/physical terminal ID to register the Natural session (refer to the SYSTP utility, User Sessions) and instead of the temporary storage key for that session.

Then this logical terminal ID is also the default value for the *INIT-ID system variable.

The NCITIDEX terminal ID exit interface gets control earlier than the NCIUIDEX/NATUEX1 user exit interface, therefore the system variable *INIT-ID set by the NCITIDEX exit interface can be modified once more by the NCIUIDEX/NATUEX1 user exit interface.

This is important to know when you are running Natural transactions under a CICS session manager.

Restrictions

1. The Natural terminal ID fields are 8 characters long. So the NCITIDEX terminal ID exit may set an 8-character terminal ID. However, the CICS terminal IDs are just 4 characters long and they are unique per CICS region. Therefore as the Natural CICS interface NCI takes only the first 4 characters of the logical terminal ID for its purposes, it lies in the user's responsibility to make sure that all logical terminal IDs are unique regarding the first 4 characters.
2. Certain Natural CICS interface functions cannot work if the first 4 characters of the logical terminal ID do not match the physical terminal.

As a consequence,

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

NCIUIDEX - User ID Exit Interface

Natural provides a user exit interface to determine whether or not a user is authorized to use Natural. The name of this user exit is NATUEX1.

NATUEX1 - User Exit

NATUEX1 is called using standard calling conventions (registers 13, 14, 15 and 1) whenever a Natural user session is activated (see the following section).

For the CICS environment, the standard calling conventions are not sufficient to issue CICS request calls and to obtain addressability of CICS control blocks. Therefore, the load module NCIUEX1 is delivered as an interface. This module calls the user exit NCIUIDEX (formerly CMUIDEX) using the standard linkage conventions, but in addition passing CICS related addresses into other registers: R6 (TCTTE), R4 (EIB), R5 (EISTG).

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 must be reassembled and linked.

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.
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 NSxx, NLxx, NRxx or NUSnnnn 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,", the Natural CICS interface will always setup an asynchronous Natural session, regardless of whether the session is terminal-bound or not.

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

Natural CICS Generation Parameters

This part of the Natural CICS Interface documentation describes the Natural CICS generation parameters. It covers the following topics:

- NCISCPCB Generation Parameters
 - NCMDIR Macro Parameters
 - NCMTGD Macro Parameters
 - NTSWPRM Macro Parameters
 - NCIPAPM Generation Parameters
 - NCMPRM Macro Parameters
-

References to CICS Tables

Where appropriate, any references to CICS tables (DCT, FCT, PCT, PPT, TCT, TST, etc.) can be considered as references to the corresponding:

- assembly-type resource definitions,
- online resource definitions via CEDA,
- batch resource definitions via DFHCSDUP.

Related Documents

- **Installation** - refer to Installing the Natural CICS Interface in the Natural Installation Guide for Mainframes.
 - **Utility** - refer to the Natural utility SYSTP which provides various TP-monitor-specific functions
 - **Operation, Individual Components** - for information on operation and the individual components of Natural in a CICS environment, see also:
Node Error Program Considerations for Natural | CICS 3270 Bridge Considerations | 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 Installing the Natural CICS Interface, Create the System Directory (in the Natural Installation Guide for Mainframes).

NCISCPCB contains the following macros:

- NCMDIR
- NCMTGD
- NTSWPRM

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 NCISPCB 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 | TSKEY | TSRECSZ | USERS

CICSPLX - Switching of CICS Application Region

This parameter is applicable under OS/390 and z/OS only.

CICSPLX=YES	If YES is specified, the 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 ADAMODE profile parameter to be set to greater than 0.
CICSPLX=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.
CICSPLX=NO	Vital Natural session data is kept in the local CICS AOR, which in fact disables CICS AOR switching.

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, i.e. 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', i.e. 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, the SUBSID parameter value in effect for the Natural session initializing the NCI environment will be taken.



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

ROLLSRV - Roll Server Rolling

This parameter is applicable under OS/390 and z/OS only.

ROLLSRV=NO	This is the default setting, if CICSPLX=NO and SIPSERV=NO. If CICSPLX or SIPSERV is YES, ROLLSRV=YES is forced.
ROLLSRV=YES	Specifying YES causes the 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), the SUBSID parameter value in effect for the Natural session initializing the NCI environment will be taken.

SIPSERV - Authorized Services Manager's Session Information Pool

This parameter is applicable under OS/390 and z/OS only.

SIPSERV=NO	This is the default setting, if CICSPLX=NO. If CICSPLX is not NO, SIPSERV=YES is forced.
SIPSERV=YES	Causes the 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), the SUBSID parameter value in effect for the Natural session initializing the NCI environment will be taken.



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

TSKEY - Prefixes for Natural CICS Temporary Storage Key

The TSKEY parameter defines the constant prefix of the temporary storage queue (see explanation below). Possible values are:

TSKEY=(xxxx,yyyy)	xxxx defines the prefix for roll data, yyyy defines the prefix for pseudo-conversational restart data.
TSKEY=(NAT2,NCOM)	This is the default setting.

When CICS temporary storage (main or auxiliary) is to be used for the Natural CICS interface roll facility or for the communication area for pseudo-conversational Natural tasks (as described with the NCMPRM macro parameter COMARET), 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.

The 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

The TSRECSZ parameter defines the maximum record length for rolling of data if CICS temporary storage is to be used as Natural CICS interface roll facility. A value specification of MAX for *nnnnn* and *mmmmm* sets the maximum value which is possible in CICS. Possible values are:

TSRECSZ=(<i>nnnnn,mmmmm</i>)	<p>The first subparameter "<i>nnnnn</i>" applies to CICS main temporary storage and must be in the range of 4032 to 32763 or 0; if it is set to 0, CICS main temporary storage cannot be used for a Natural roll facility.</p> <p>The second subparameter "<i>mmmmm</i>" applies to CICS auxiliary temporary storage and must be in the range of 3976 to 32763 or 0; if non-zero, this value is used unconditionally; if set to 0, the 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.</p> <p>A user-defined record size greater than CI size results in fewer (logical) roll I/Os at the expense of additional CICS overhead due to writing spanned records.</p>
TSRECSZ=(32748,0)	This is the default setting.

USERS - Session Information Record

This parameter specifies the number of session information record slots (SIRs). Possible values are:

USERS=(<i>nnnnn,mmm</i>)	<p>The subparameter "<i>nnnnn</i>" defines the number of SIRs to be held in the Natural CICS directory module itself. "<i>nnnnn</i>" must be in the range from 1 to 32767. When the SIR slots in the directory are occupied, the Natural CICS interface acquires a CICS shared storage segment, large enough to hold the number of SIRs defined by "<i>mmm</i>", which must be in the range from 0 to 255.</p> <p>If the subparameter "<i>mmm</i>" 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.</p> <p>If a value other than 0 is specified for "<i>mmm</i>", secondary storage segments are allocated automatically as required. Allocated secondary segments are freed again if they are no longer needed.</p>
USERS=(100,20)	This is the default setting.

The 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 Parallel Sysplex environment;
- in a data space of the Natural Authorized Services Manager when running in multiple CICS regions inside a single OS/390 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. The 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.

PFKEY=xxx	Possible values for xxx are: PF1 to PF24, PA1 to PA3. Also list of keys can be specified. No default value is provided.
PFKEY=(xxx,xxx,...)	Also a list of keys can be specified. This has to be enclosed in parantheses, e.g. PFKEY=(PF12, PF14).

When starting a session, the 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.



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

The PRIMERF 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:

PRIMERF=VSAM	The Natural CICS interface VSAM RRDS roll files are taken as the primary roll facility. CICS auxiliary temporary storage is considered as the secondary roll facility, which means that it is used if all primary roll files become full or unavailable.
PRIMERF=AUX	CICS auxiliary temporary storage is taken as primary roll facility of the Natural CICS interface.
PRIMERF=MAIN	CICS main temporary storage is taken as Natural CICS interface primary roll facility.
PRIMERF=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.

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 taken.

Points to be observed:

- PRIMERF=VSAM and PRIMERF=AUX have the same effect, when no VSAM RRDS roll file is available in the CICS system.
- PRIMERF=AUX and PRIMERF=MAIN have the same effect, when auxiliary temporary storage is not defined in the CICS system.
- If auxiliary temporary storage is not defined in the CICS system, a specification of PRIMERF=VSAM implies that CICS main temporary storage is considered as secondary roll facility, in case the VSAM RRDS roll files become unavailable or full.
- If CICS main temporary storage is to be used as roll facility, the record size is defined by the TSRECSZ parameter.



Note that sessions that are associated with thread groups defined with PRIMERF=NONE cannot roll due to the lack of a roll facility and are therefore conversational by design.

THRDSIZE - Thread Size

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

THRDSIZE= <i>nnn</i>	The thread size <i>nnn</i> can be equal to 40 or greater. No default value is provided.
----------------------	---

Note that this parameter defines the **logical** thread size that is available to Natural. However, the Natural CICS interface NCI 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 THRDSIZE 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, 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:

THREADS= <i>nnn</i>	The number of threads can be equal to 999 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 the 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 queuing 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=SETM/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

The TRAN parameter defines a single CICS transaction or a list of them.

TRAN=	Possible values: 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, the 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 parentheses, e.g. TRAN=(NATU, XYZ).



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:

TYPE=SHR (default value)	<p>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.</p> <p>When running in a SYSplex environment, the Natural parameter RELO=OFF forces sessions with TYPE=SHR threads to be conversational to prevent a CICS region switch.</p>
TYPE=GETM	<p>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.</p> <p>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.</p>
TYPE=NONE	<p>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.</p>
TYPE=ALIAS	<p>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.</p>

XTRAN - Hexadecimal Transaction IDs for Thread Group

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

XTRAN=	Possible values: one or more CICS transaction codes defined in the PCT for Natural.
--------	---

No default value is provided.



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.

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

NTSWPRM Macro Parameters

The NTSWPRM macro is used to define the various aspects of the swap pool. If no swap pool is to be used, omit this macro. For more information, see Natural Swap Pool in the Natural Operations for Mainframes documentation.

NCIPAPM Generation Parameters

The Natural CICS interface parameter module is generated by assembling the NCIPARM source module, see Installing the Natural CICS Interface, Create the Natural CICS Parameter Module (in the Natural Installation Guide for Mainframes). It holds the NCMPRM macro definition.

The purpose of the NCMPRM macro and the parameters which can be specified in it are described in the following section.

NCMPRM Macro Parameters

The macro NCMPRM determines all Natural session options that are relevant in a CICS environment. This macro is part of the Natural CICS parameter module, which is created in the corresponding step of the Natural under CICS Installation Procedure in the Natural Installation Guide for Mainframes.

A sample NCMPRM macro definition, including all default settings, is contained in the NCIPARM source module in dataset NCInnn.SRCE.

The individual parameters of the NCMPRM macro are described below.

BACKEND | BACKOUT | CHAP | COMACAL | COMAMSG | COMARET | CONSOLE | DUPTID | FLDLEN | LOGDEST | MSGDEST | MSGPFX | MSGTRAN | PREFIX | PSTRNID | RESENDC | RESENDS | RJEDEST | RJEUSER | SIGNON | SLCALL | SNDLAST | TERMVAR | TRANCHK | UCTRAN | UNITID

BACKEND - Back-End Program Invocation Control

The BACKEND parameter defines whether a specified back-end program or transaction is to be invoked after the session has terminated (normally or abnormally).

The BACKEND parameter has two sub-parameters. The second sub-parameter is optional. It controls if a back-end program is to be invoked in the event of a terminal error. This also includes session clean-up tasks started by NEP.

Possible values are YES/NO for both sub-parameters, but the default values are different.

BACKEND=YES	Same as BACKEND=(YES,NO). This is the default if the BACKEND parameter is omitted. A potential back-end program or transaction is always invoked, particularly after task abends, but not in the case of terminal errors. When a back-end program is invoked, the Natural termination message and return code are passed to the CICS transaction work area (TWA). In addition, the same information can be passed to a CICS COMMAREA, as described with the COMAMSG parameter .
BACKEND=(YES,YES)	Same as BACKEND=(,YES). A potential backend program or transaction is always invoked, particularly after abends including terminal errors.
BACKEND=NO	Forces BACKEND=(NO,NO). A potential back-end program or transaction is only invoked if the Natural session has been terminated normally; that is, with a Natural termination message.

BACKOUT - Backout Transaction in the Case of Unrecoverable Abends

The BACKOUT parameter defines whether the Natural CICS interface is to perform a transaction backout by means of an EXEC CICS SYNCPOINT ROLLBACK call or not. Possible values are:

BACKOUT=YES	All pending file updates are backed out. This is the default setting.
BACKOUT=NO	All pending file updates are committed.

Because of its abnormal termination exit, the Natural CICS interface intercepts all abends. If an abend is not recoverable, all resources of the abending session are released and the session is terminated via EXEC CICS RETURN; that is, it is terminated "normally" in terms of CICS. Thus, at the end of the task, "pending" file updates are not automatically backed out by CICS.

CHAP - Change Task's Dispatching Priority

The CHAP parameter defines how the Natural CICS interface is to treat long-running tasks reaching the DBROLL and/or MAXROLL call limits. Possible values are:

CHAP=YES	The task's dispatching priority is decremented by 1 every time it reaches the DBROLL and/or MAXROLL call limits. The original task dispatching priority is re-established at the next screen I/O.
CHAP=NO	The session is suspended. This is the default setting.

COMACAL - CICS COMMAREA Usage for Subroutine Calls

The COMACAL parameter defines whether the Natural CICS interface is to take advantage of the CICS command level COMMAREA facility when invoking external subroutine programs with EXEC CICS LINK. Possible values are:

COMACAL=YES (default value)	The Natural parameter list addresses are passed in both the CICS TWA (as with Natural Version 1) and in a CICS COMMAREA (as with Natural Version 2). The COMMAREA length setting is 8 or 12, depending on the FLDLEN parameter's setting. Specifying YES in particular improves the communication with external subroutine programs written in CICS command level, because it is easier for these subroutines to access a CICS COMMAREA than a CICS TWA; it also results in less overhead.
COMACAL=NO	Forces Natural to pass the Natural request parameter list address to an external subroutine program in the CICS TWA only. The COMMAREA length setting is 0.

Actually, the COMACAL parameter can provide "compatibility mode" to Natural Version 1 regarding the way of passing data to called external subroutines.

Set this parameter to YES if you want to take advantage of the CALL option SET CONTROL 'P=C'; see also the description of the terminal command "%P" in the Terminal Commands documentation.

COMAMSG - CICS COMMAREA Usage for Termination Messages

The COMAMSG parameter controls whether to pass the Natural termination message and return code to a potential back-end program or transaction in a CICS COMMAREA. Possible values are:

COMAMSG=YES	The Natural back-end parameter area and potential termination data are passed in a CICS COMMAREA. This is the default setting.
COMAMSG=NO	This setting forces Natural at session termination (normal or abnormal) to pass the Natural back-end parameter area (see also Back-End Program Calling Conventions in the Natural Operations for Mainframes documentation) to a potential back-end program in the CICS TWA only. Potential termination data are still passed in the COMMAREA. If there are no termination data available, no COMMAREA is passed.

COMARET - CICS COMMAREA Usage for Task Control

The COMARET parameter defines whether the Natural CICS interface is to take advantage of the CICS command level COMMAREA facility when terminating and restarting pseudo- conversational tasks.

COMARET=YES	A pseudo-conversational Natural task saves its restart information into a CICS COMMAREA, unless it has been invoked with EXEC CICS LINK or the equivalent CICS macro request. This is the default setting.
COMARET=NO	Forces Natural to place its restart information into CICS main temporary storage, which results in more overhead because of additional CICS service calls necessary to place and retrieve this information. The CICS temporary storage key used consists of a prefix string (as defined with the NCMDIR parameter TSKEY and of the terminal ID. If running in a CICSplex environment, the CICS temporary storage key prefix must be defined in a CICS TST as REMOTE/SHARED to be accessible in all participating CICS regions.

Actually the COMARET parameter can provide compatibility to Natural Version 1 in terms of where to put pseudo-conversational restart data.

CONSOLE - CICS Terminal ID for the Operator Console

The CONSOLE parameter specifies the terminal ID used by Natural for message switching to the operator console. Possible values are:

CONSOLE=xxxx	xxxx can be any 4-character terminal ID.
CONSOLE=CN01	This is the default setting under OS/390.
CONSOLE=CNSL	This is the default setting under VSE/ESA.

The console terminal must allow automatic transaction initiation (ATI) in the terminal entry.

DUPTID - Handle Duplicate Terminal ID

The Natural CICS interface requires unique terminal IDs, because the terminal ID is the key for its session information records (SIRs). This is normally guaranteed for a single CICS region, but not necessarily over several CICS regions sharing the same SIP server.

The DUPTID parameter determines how the Natural CICS interface has to deal with duplicate terminal IDs, that is, when a new session is to be started and an SIR already exists for this terminal ID.

DUPTID=YES	This is the default setting. If a duplicate terminal ID is encountered, the Natural CICS interface internally forces the old session to terminate and, after that, starts a new session.
DUPTID=NO	When a SIR already exists for the new session's terminal ID, the Natural CICS interface terminates the new session and issues the message NS19.

A terminal ID exit interface is available to create unique 8-character terminal IDs, for example, by appending the 4-character CICS system ID to the physical 4-character CICS terminal ID, which results in a logical Natural terminal ID.

FLDLEN - Supply Field Length List On External Program Call

The FLDLEN parameter defines whether the field length list addresses are to be passed or not when invoking external subroutines via EXEC CICS LINK.

FLDLEN=NO (default value)	Same as FLDLEN=(NO,NO) Only the parameter address list address and the field description list address (R1 and R2, as described with the CALL statement) are passed in the CICS TWA and in a CICS COMMAREA, unless the COMACAL is set to NO.
FLDLEN=YES	Same as FLDLEN=(YES,NO) The field length list address (R3, as described with the CALL statement) is passed in addition in a CICS TWA and in a COMMAREA, respectively.
FLDLEN=(YES,YES)	Same as FLDLEN=(,YES) The field length list address and the large field length list address (R4, as described with the CALL statement) are passed in addition in a CICS TWA and in a COMMAREA, respectively. The second subparameter set to YES forces a YES condition for the field subparameter.

The following table lists the possible combinations of COMACAL and FLDLEN and their associated CALL parameter values:

NCIPARM Parameters		CALL Parameters		
COMACAL	FLDLEN	TWA parameter addresses	COMMAREA parameter addresses	COMMAREA length
NO	NO,NO	2	n/a	0
NO	YES,NO	3	n/a	0
NO	YES,YES	4	n/a	0
YES	NO,NO	2	2	8
YES	YES,NO	3	3	12
YES	YES,YES	4	4	16

In any case, the last address passed gets a flag saying it is the last address in the list. This flag is set in the high order bit in the address field.

LOGDEST - Natural CICS Logging Destination

This parameter specifies the name of a CICS destination, where the Natural CICS interface writes its session log records to. Possible values are:

LOGDEST=	Any valid destination name
LOGDEST=NLOG	This is the default setting.

A CICS destination control table entry must be defined for the optional Natural CICS log dataset.

MSGDEST - Destination ID for Natural Error Message Logging

This parameter specifies the name of the CICS destination to be used by the Natural CICS interface to log the Natural session termination message if a session terminates abnormally.

MSGDEST=	Any valid destination name
MSGDEST=NERR	This is the default setting.

Since these messages are in character format, any already available CICS destination (for example, CSSL) can be used rather than defining a new one.

MSGPFX - Generate NCI Message Prefix for WTL Messages

The Natural CICS interface uses a prefix for all messages it sends to the MSGDEST destination. This prefix has a length of approximately 48 bytes and comprises the following information:

- NCI message number,
- CICS region SYSID,
- terminal ID or the string ASYN for non-terminal tasks,
- user ID,
- transaction ID,
- date and time.

By default, the message prefix is also appended to those messages which are output through CMWTL.

MSGPFX=YES	The NCI message prefix is appended to all messages which are issued through CMWTL. This is the default setting.
MSGPFX=NO	The NCI message prefix is not appended to the messages which are issued through CMWTL. The messages are issued unchanged.

MSGTRAN - Internal Message Switching Transaction ID

The parameter specifies the transaction ID internally used by the Natural message switching and asynchronous session flushing facilities.

MSGTRAN=	Any valid CICS transaction ID.
MSGTRAN=NMSG	This is the default setting.

This transaction ID must be different from any transaction ID used to invoke Natural, and it must be defined in CICS.

PREFIX - Common Prefix for Programs and Files

This parameter defines a common module *prefix* for the Natural CICS components as the Natural CICS system directory *prefix* CB, the CICS 3270 Bridge XFAINTU exit *prefix* XFA, the VSAM roll files *prefix* Rn, n=1 - 9, and system control records in CICS main temporary storage holding information about all permanent GETMAIN storages by NCI as local pools and shared threads. The TS control record keys are of the form *prefix* X CR, where X is an unprintable character.

PREFIX= <i>prefix</i>	<i>prefix</i> can be 1 to 5 bytes long and must conform to the naming conventions for programs and files. No default value provided.
-----------------------	---

PSTRNID - Control of *INIT-PROGRAM Variable Setting

When a Natural task is activated by a front-end program, the PSTRNID parameter determines, how the Natural variable *INIT-PROGRAM is set. Possible values are:

PSTRNID=YES	*INIT-PROGRAM is set to the actual transaction ID used for Natural CICS pseudo-conversational task processing, which is not necessarily the transaction ID of the task which originally started the Natural session. This is the default setting.
PSTRNID=NO	*INIT-PROGRAM is set to the transaction ID of the task, which originally started the Natural session.

RESENDC - Check for Screen Re-sending after Subroutine Calls

Natural optimizes the 3270 output data stream by default. The screen imaging technique used by Natural makes it possible for Natural to always remember the map most recently sent. Thus, when sending a new map, Natural actually sends "updates" of the old map only. With this logic, a screen image can get destroyed by 3GL programs called by Natural which perform screen I/Os themselves.

RESENDC=YES (default value)	The Natural CICS interface checks whether any called 3GL programs have performed screen I/Os. If so, the Natural CICS interface causes Natural to send a full screen with the next screen I/O.
RESENDC=NO	The Natural CICS interface causes Natural to send only updates.

RESENDS - Screen Re-send Check after Pseudo-Conversational Session Resume

Natural optimizes the 3270 output data stream by default. The screen imaging technique used by Natural makes it possible that Natural always remembers the map most recently sent. Thus, Natural only sends "updates" when sending a new map, too. With this logic a screen image can get destroyed, for example, by message switching (CICS CMSG transaction) during pseudo-conversational screen I/O.

RESENDS=YES (default value)	During the Natural session, the Natural CICS interface also recognizes screen I/Os from outside and causes Natural to re-send the screen most recently issued.
RESENDS=NO	Natural only sends "updates" when sending a new map.

RJEDEST - Name of the Natural CICS Submit Destination

The parameter applies to OS/390-type operating systems only.

RJEDEST=	Destination name.
RJEDEST=NRJE	This is the default setting.

RJEDEST specifies the *destination name* of the CICS extra partition destination used by the NATRJE utility for submitting jobs via the JES internal reader facility.



An appropriate CICS destination must be defined in the CICS DCT and start-up JCL; see also the corresponding step of the **Installation Procedure for the Natural CICS Interface** (in the Natural Installation Guide for Mainframes).

Function code "L" or "B" (*parm3* of the NATRJE CALL statement) must be set for the last NATRJE call.

- ^L When "L" is specified and *nrje* is an extra partition destination, the destination is closed, which in turn triggers the start of the internal reader.
- ^B When "B" is specified and *nrje* is an indirect destination, the destination is not closed; in this case, a trailing "/*EOF" card must be submitted in order to trigger the start of the internal reader.

For further information on the Natural NATRJE utility, refer to the Natural Utilities documentation.

RJEUSER - Submit to POWER User ID Setting

The RJEUSER parameter only applies to VSE/ESA operating systems using the POWER spooling system.

RJEUSER=YES (default value)	Sames as RJEUSER=(YES,CICS). The Natural system variable *INIT-USER is used as the XPCC user ID and the POWER JECL must be set up appropriately by the user.
RJEUSER=(YES,NAT)	The Natural system variable *USER is used as the XPCC user ID and the POWER JECL must be set up appropriately by the user.
RJEUSER=NO	The user ID 'R000' is used as the XPCC user ID for all jobs submitted by the Natural CICS interface.

In VSE/ESA operating systems, Natural under CICS performs job submission by means of XPCC macro requests.

The XPCC macro requires the specification of a user ID, thus giving access to the submitted job's list or punch output to the submitting user only, unless appropriate LDEST/PDEST parameters have been specified in the * \$\$ JOB statement or appropriate DEST parameters have been specified in the * \$\$ LST or * \$\$ PUN statement respectively.

Using the special user ID 'R000', however, gives common access to list or punch output of a submitted job without having to code appropriate target destinations in the JECL.

SIGNON - SIGNON Behavior

This parameter defines how Natural under CICS should deal with a CICS user ID for a Natural session.

SIGNON=NO (default value)	Natural under CICS always does an EXEC CICS ASSIGN USERID (..); when users have not signed on to CICS via CESN/CSSN, recent CICSes will return the CICS default user ID.
SIGNON=YES	Natural under CICS only does an EXEC CICS ASSIGN USERID (..) <ul style="list-style-type: none"> a) if it is a terminal task, b) if the user has signed on to CICS.

Further processing:

Any non-blank result of EXEC CICS ASSIGN USER ID (..), if executed, is accepted for Natural *INIT-USER ID.

If blank, the edited (unpacked) CICS task number is taken instead for asynchronous CICS sessions. For terminal-bound tasks, the CICS 3-byte operator ID is taken when it is non-blank, otherwise the CICS terminal ID is taken for Natural *INIT-USER ID.

Notes:

1. CICS terminal IDs are unique within a CICS region, while CICS user and operator are not necessarily. However, CICS terminal IDs may have duplicates in other CICS regions resulting in duplicate user IDs in Adabas.
2. Natural user ID exit NATUEX1 or Natural CICS user ID exit interface NCIUIDEX may be used to customize *INIT-USER.

SLCALL - Standard Linkage Call

The Natural CALL statement invokes a dynamic non-Natural program using CICS conventions, that is, via an EXEC CICS LINK. A dynamic non-Natural program can also be invoked with standard linkage conventions (for example BALR/BASR/BASSM 14,15) if an appropriate indicator is set in the Natural program before the CALL statement is executed (see also the terminal command %P=S).



The terminal command %P=S bypasses the SLCALL automatism of using a certain linkage convention.

SLCALL enables you to automatically use a certain linkage convention. This is particularly relevant in CICS systems where the CICS macro level API is no longer supported, which is the case in CICS/ESA Version 3.2 or above. Possible values are:

SLCALL=YES	The Natural CICS interface determines whether the module to be called is a valid CICS command level program by looking for the string "DFH" at the module's load point. If "DFH" is found, the program is invoked via an EXEC CICS LINK. If "DFH" is not found, the module is treated according to standard linkage conventions and is invoked via BALR/BASSM 14,15.
SLCALL=NO	The linkage convention is not used. This is the default setting.

SNDLAST - LAST Option Usage for EXEC CICS SEND Commands

The SNDLAST parameter is useful for SNA terminals (LUTYPE2) with bracket protocol to force "end bracket" for pseudo-conversational screen I/Os.

SNDLAST=YES (default value)	The LAST option is used for EXEC CICS SEND commands before the task terminates in pseudo-conversational mode.
SNDLAST=NO	The LAST option is not used.

TERMVAR - Terminal ID Variable for Natural Work Files

This parameter enables a Natural user to have exclusive Natural work files under CICS without having to know the terminal ID.

TERMVAR=xxxx	Variable xxxx is a four-character string. See explanation below.
TERMVAR=&TID	This is the default setting.

As terminal IDs are unique in a CICS session, exclusive work files in CICS temporary storage usually contain the CICS terminal ID. TERMVAR allows you to define a variable. If this variable is found in a work file name, it will be replaced by the actual terminal ID. Strings with non-alphanumeric characters must be enclosed in apostrophes (' ').



The variable string must not contain the substring '***', because Natural will replace this substring with the work file number, which makes it impossible to insert the terminal ID.

TRANCHK - Check Input Map for Transaction ID

If a connection to a CICS session gets lost or dropped (for example under VM or when a session manager is installed) without having terminated the session, another user can get into this open session when calling CICS. Usually, the first action of a user in a CICS environment is to enter a transaction ID. This parameter offers the following options:

TRANCHK=YES	The Natural CICS interface checks whether the first 4 bytes of the transaction ID entered by the user matches the Natural transaction ID. If so, the Natural CICS interface assumes a "restart" after a connection has been lost or dropped. All resources of the "old" session are freed and a new session is started.
TRANCHK=NO	Data entered by the user are not checked for the Natural transaction ID. This is the default setting.

UCTRAN - Lower/Mixed Case Support in Natural

This parameter enables or disables the lower/mixed case support by the Natural CICS interface (NCI). Possible values are:

UCTRAN=YES (default value)	Same as UCTRAN=(YES,YES). NCI lower/mixed case support is fully enabled.
UCTRAN=NO	Same as UCTRAN=(NO,YES). NCI lower/mixed case support is disabled for pseudo-conversational screen I/Os.

The first subparameter controls NCI mixed case support after a conversational screen I/O, while the second subparameter controls NCI mixed case support after a conversational screen I/O; the latter also includes NTC uploads.

First Subparameter (pseudo-conversational screen I/Os)

To accomplish lower/mixed case support for pseudo-conversational Natural sessions, it is necessary that the terminal input be not already translated to upper case before the Natural nucleus gets control. Therefore the Natural CICS interface by default switches terminals defined with UCTRAN(YES) into mixed mode (UCTRAN(TRANID)) for the lifetime of the Natural session.

As for security reasons any modification of CICS definitions / control blocks may not be desired, the Natural CICS interface can be prevented from modifying a terminal's upper case translation status by setting this NCIPARM UCTRAN parameter to NO. If so, the user must define a terminal as running in "lower case" (CICS TYPETERM parameter UCTRAN(TRANID/NO)) to be able to use the Natural lower/mixed case support. As all CICS versions supported by the current Natural Version provide "case switching" on transaction level via the UCTRAN parameter in a transaction's PROFILE, this NCIPARM parameter should be set to NO, thus leaving lower/mixed case support to CICS.

Note:

In CICS, the combination of the UCTRAN parameters in both TYPETERM and PROFILE definitions determine how CICS treats the terminal input of a pseudo-conversational transaction (for details see CICS Resource Definition Manual or others). Therefore it is always advisable that mainly the PROFILE associated to a transaction defines the required upper case translation status thus making an application unaffected by any TYPETERM upper case translation mode changes.

Second Subparameter (conversational screen I/O)

Lower/mixed case support for conversational I/Os means that the Natural CICS interface uses the "as is" option on the CICS terminal input requests (CONVERSE/RECEIVE ASIS). If the second subparameter is set to NO, the Natural CICS interface does the conversational CICS terminal input requests without the "as is" option,

UNITID - Establish Unique Terminal IDs

UNITID=YES	The Natural CICS Interface appends a CICS system ID (local SYSID if no MRO, otherwise TOR SYSID) to the 4-byte CICS terminal ID, thus creating an 8-byte logical terminal ID.
UNITID=NO	The Natural CICS Interface uses the CICS terminal ID as it is. This is the default setting.

This parameter is of interest, when resources are shared as SIP server or roll server by several CICS regions, particularly in non-CICSplex: If the same terminal IDs are used in several CICS environments, this parameter helps to provide unique terminal IDs for Natural. Inside the Natural CICS interface, Natural terminal IDs are 8-byte fields, and a combination of 8-byte terminal ID and 8-byte CICS user ID is taken as key for SIP and the roll server.

The result of this parameter is used by the Natural CICS Interface for the session key and the roll server key and by Natural for the system variable *INIT-ID.

Note that a terminal ID exit (NCITIDEX) possibly will post-process that logical terminal ID. Also note that a user ID exit (NCIUIDEX and NATUEX1) may post-process the *INIT-ID system variable.

NCISCPRI Warnings and Error Messages

This part of the Natural CICS Interface documentation describes the NCISCPRI Warnings and Error Messages.

References to CICS Tables

Where appropriate, any references to CICS tables (DCT, FCT, PCT, PPT, TCT, etc.) can be considered as references to the corresponding:

- assembly-type resource definitions,
- online resource definitions via CEDA,
- batch resource definitions via DFHCSDUP.

Related Documents

- **Installation** - refer to Installing the Natural CICS Interface in the Natural Installation Guide for Mainframes.
 - **Utility** - refer to the Natural utility SYSTP which provides various TP-monitor-specific functions
 - **Operation, Individual Components** - for information on operation and the individual components of Natural in a CICS environment, see also:
Node Error Program Considerations for Natural | CICS 3270 Bridge Considerations | 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 Work Files Under CICS
-

NCISCPRI Warnings and Error Messages

The following messages may be output during the initialization of VSAM roll files for Natural; see also the corresponding step of the Installation Procedure for the Natural CICS Interface in the Natural Installation Guide for Mainframes:

mmmmmmmm REQUEST FAILURE AT OFFSET xxxxx, RTC=rrrr, ERROR=eeee, FTNCD=ffff (E)

A VSAM macro request failed. *mmmmmmmm* is the failing macro; *xxxxx* is the offset of the macro request within NCISCPRI; *rrrr*, *eeee* and *ffff* are the VSAM macro request return, feedback and function code.

RECORD SIZE IS LESS THAN OPTIMUM OF CI-SIZE - 7 (W)

For an (unblocked) VSAM file, the maximum possible record size is the control interval size minus 7. Any smaller record size means wasted disk space and can result in more roll I/Os.

ACCESSED FILE IS NOT AN RRDS FILE (E)

The VSAM file to be initialized is not an RRDS, but a KSDS or ESDS file. Only VSAM RRDS files are supported as VSAM roll files for Natural.

RECORD SIZE IS LESS THAN ALLOWED MINIMUM OF 4089 (E)

The minimum VSAM control interval size supported by the Natural CICS interface is 4096 and the minimum record size supported by Natural under CICS is the corresponding optimum record size.

PARAMETER INPUT OBSOLETE AND THEREFORE IGNORED (W)

NCISCPRI does not require parameter input in JCL stream. But if available, it is retrieved (until the end-of-data). The message is not issued for null files.

Customizing VSAM RRDS Roll Files

This part of the Natural CICS Interface documentation describes the customization of VSAM RRDS roll files. It covers the following topics:

- Increasing the Number of VSAM RRDS Roll Files
- Decreasing the Number of VSAM RRDS Roll Files
- Changing the Characteristics of the VSAM RRDS Roll Files

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

References to CICS Tables

Where appropriate, any references to CICS tables (DCT, FCT, PCT, PPT, TCT, etc.) can be considered as references to the corresponding:

- assembly-type resource definitions,
- online resource definitions via CEDA,
- batch resource definitions via DFHCSDUP.

Related Documents

- **Installation** - refer to Installing the Natural CICS Interface in the Natural Installation Guide for Mainframes.
 - **Utility** - refer to the Natural utility SYSTP which provides various TP-monitor-specific functions
 - **Operation, Individual Components** - for information on operation and the individual components of Natural in a CICS environment, see also:
Node Error Program Considerations for Natural | CICS 3270 Bridge Considerations | 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 Work Files Under CICS
-

Increasing the Number of VSAM RRDS Roll Files

Up to 9 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 dataset 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 the 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 the 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.

Natural in CICS MRO Environments

This part of the Natural CICS Interface documentation describes the functionality of Natural in CICS Multi-Region (MRO) Environments. It covers the following topics:

- NCIPARM Parameter COMARET Set to YES
 - NCIPARM Parameter COMARET Set to NO
-

References to CICS Tables

Where appropriate, any references to CICS tables (DCT, FCT, PCT, PPT, TCT, etc.) can be considered as references to the corresponding:

- assembly-type resource definitions,
- online resource definitions via CEDA,
- batch resource definitions via DFHCSDUP.

Related Documents

- **Installation** - refer to Installing the Natural CICS Interface in the Natural Installation Guide for Mainframes.
- **Utility** - refer to the Natural utility SYSTP which provides various TP-monitor-specific functions
- **Operation, Individual Components** - for information on operation and the individual components of Natural in a CICS environment, see also:
Node Error Program Considerations for Natural | CICS 3270 Bridge Considerations | 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 Work Files Under CICS

Special considerations apply when running Natural in a CICS multi-region (MRO) environment.

NCIPARM Parameter COMARET Set to YES

When COMARET is set to YES, 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, swap pool, 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.

NCIPARM Parameter COMARET Set to NO

When COMARET is set to NO, 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 Considerations for Natural and Natural CICS Sample Programs), which terminates a Natural session when the associated terminal is removed.

CICS Node Error Program Considerations for Natural

This section discusses CICS node error program considerations. The following topics are covered:

- Normal Situation
- Situations Not under Control of Natural CICS Interface
- Recovery Mechanisms
- Special Considerations
- Example Dummy Program

See also:

- Installing a CICS Node Error Program
For information on installing a CICS node error program, refer to the section concerning user-replaceable programs in the **CICS Customization Guide** for your CICS version.
-

Normal Situation

Whenever a Natural session is active, CICS resources such as thread storage, roll facility entries (that is, records in a VSAM RRDS file or in a CICS temporary storage queue), swap pool slots etc. are used.

If these resources are under the control of the Natural CICS interface, they are correctly released whenever a session terminates normally or abnormally.

Situations Not under Control of Natural CICS Interface

The following cases cannot be controlled by the Natural CICS interface:

1. A non-Natural program called by Natural issues an EXEC CICS ABEND CANCEL command or the equivalent CICS macro request: the CICS task is canceled without the Natural CICS interface receiving control to properly release all session resources.
2. Some CICS monitor products offer tools to purge CICS tasks, bypassing any abnormal termination exit set by the application. If a Natural task is canceled this way, the Natural CICS interface has no chance to release the resources still owned by the session.
3. A user disconnects a terminal from the CICS region (by switching the power off or using an adequate session manager function) while a Natural session is currently not active in CICS (pseudo-conversational screen I/O).

Recovery Mechanisms

The Natural CICS interface provides some recovery mechanisms to recover from such situations; for example:

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

However, it may take quite a long time between logically terminating the session and releasing its resources, and there may also be a security concern:

When the NCIPARM generation parameter COMARET is set to NO, the information to resume a Natural session is kept in a CICS temporary storage record with the terminal ID being part of the temporary storage queue name. If another CICS user tries to start Natural with this terminal ID, he/she will resume the old Natural session rather than starting a new one.

The third case in the above list is the most crucial one. CICS provides a node error program (NEP) exit interface, which can be used in these cases to trigger the Natural CICS interface to terminate the lost session. An appropriate program called NCIZNEP is provided in the Natural CICS source library (see Natural CICS Sample Programs); it must be called by a DFHZNEP node error program.

Special Considerations

There are still some items to be considered:

- With CICS versions prior to CICS/ESA 3.2, the CICS node error program had to be written using CICS macro level calls, whereas with CICS/ESA 3.2 and above, the CICS node error program must be a command level program.

Therefore, the Natural CICS source library contains two sample node error programs: XNCINEP1 for CICS versions prior to CICS/ESA 3.2, and XNCINEP2 for CICS/ESA 3.2 and above.

Both sample programs do not perform anything special for the Natural CICS environment, they merely call (via LINK) the NCIZNEP program, which then deals with Natural under CICS.

- DFHZNEP may already be customized for a specific installation; as only one node error program is possible, the logic of the relevant XNCINEP_x program should be adapted to the existing DFHZNEP logic.
- In MRO environments, DFHZNEP must be installed in the TOR.
- When you are using the CICS storage protection feature with CICS 3.3 or above, the NCIZNEP program must be defined with EXECKEY(CICS).
- In the case described under 3. above, DFHZNEP may receive control more than once for various internal error codes, since each internal error code is related to a specific CICS error message, but there may be more than one error message resulting from a given action.
- The CICS control block constellation may have changed each time a node error program has been invoked, for example, the COMMAREA and NEXTRANSID information in the TCTTE may have been lost after a certain node error event.

In this case the NCIPARM parameter COMARET must be set appropriately, which means that you cannot choose a node error event for your node error program to be invoked when passing the Natural pseudo-conversational session restart data in a CICS COMMAREA that has already been cleaned up by CICS.

Example Dummy Program

If you want to know how many times and with what error codes DFHZNEP is invoked on certain actions and how the TCTTE should look, write a dummy node error program, which only issues CICS trace requests showing the requested information.

The following sample enables a DFHZNEP error processor to receive control for all possible error codes passed to DFHZNEP:

```
.
DFHSNEP TYPE=INITIAL
ORG NEPTT
DC 256X'03' invoke error processor '3' for ALL error codes
ORG ,
```

```
DFHSNEP TYPE=ERRPROC,GROUP=3,CODE=49
```

```
.  
set up requested information and issue trace request(s)
```

CICS 3270 Bridge Support

This section of the Natural CICS Interface documentation describes the CICS 3270 Bridge support. The following topics are covered:

- Default Support of CICS 3270 Bridge
 - Full CICS 3270 Bridge Support
 - NCIXFATU - NCI Source Module
 - Profile Parameter DSC=OFF Recommended
-

Default Support of CICS 3270 Bridge

By default, the Natural CICS interface supports the CICS 3270 Bridge by being able to deal with "bridged devices", i.e. 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 source module NCIXFATU. Refer to the corresponding step Assemble the Natural/CICS Interface XFAINTU Exit of the section Installing Natural under CICS in the Natural Installation Guide for Mainframes.

NCIXFATU - NCI Source Module

The NCIXFATU module actually 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 provides the same functionality for Natural as a Node Error Program (NEP) provides for "real" terminals.

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 to force Natural always to send full screens rather than the delta to the previous screen.

Special Natural CICS Functionality

This part of the Natural CICS Interface documentation explains special Natural CICS functionality. It covers the following sections:

- Calling Non-Natural Programs
 - Dummy Screen I/Os with Natural under CICS
 - NCISTART - Natural CICS Nucleus
-

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/Os and other work load without Natural receiving control in between. If a non-Natural program is doing conversational screen I/Os, 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 CICS standard linkage conventions; that is, via BALR R14,R15.

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

Calling Non-Natural Programs with Parameter Values in a COMMAREA

By default, non-NATURAL programs are called with the addresses of the request parameter and field descriptor lists (R1 and R2; see also the description of the CALL statement in the Natural Statements documentation) passed in the TWA and COMMAREA (depending on the setting of the NCIPARM parameter COMACAL).

A more CICS-like method is to pass the parameter values in a CICS COMMAREA, 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 command %P=C in the Terminal Commands documentation.

Prerequisite: This functionality requires COMACAL to be set to YES in NCIPARM.

When the parameter values are passed to a CICS COMMAREA, the first two words of the CICS TWA are low-value, which means that no R1 and R2 information is passed.

Dummy Screen I/Os 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/Os 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.

NCISTART - Natural CICS Nucleus

NCISTART (that is, the Natural CICS nucleus with "NCISTART" as entry point) is eligible to be placed into the CICS PLTSD for CICS quiesce stage 1 or 2 execution.

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

NCISTART 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 (with CICS/ESA 3.2 or above) or at the beginning of the TWA (with earlier CICS versions).

Natural CICS Sample Programs

This part of the Natural CICS Interface documentation describes the Natural CICS sample programs. It covers the following topics:

- Sample Programs in Natural CICS Source Library
- Sample Programs for Use with VSE/ESA

You can find a more detailed explanation of all these programs in the corresponding program source itself.

Sample Programs in Natural CICS Source Library

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

XNCI3GC1 - Subprogram Call

Programming Language: COBOL

This program provides a sample COBOL call to a Natural subprogram under CICS.

XNCI3GP1 - Subprogram Call

Programming Language: PL/1

This program provides a sample PL/1 call to a Natural subprogram under CICS.

XNCIFRCL - Front-End Program

Programming Language: COBOL

This is a front-end program for invoking Natural via EXEC CICS LINK.

XNCIFRPL - Front-End Program

Programming Language: PL/1

This is a front-end program for invoking Natural via EXEC CICS LINK.

XNCIFRCS - Front-End Program

Programming Language: COBOL

This is a front-end program for invoking Natural via EXEC CICS START.

XNCIFRPS - Front-End Program

Programming Language: PL/1

This is a front-end program for invoking Natural via EXEC CICS START.

XNCIFRCX - Front-End Program

Programming Language: COBOL

This is a front-end program for invoking Natural via EXEC CICS XCTL.

XNCIFRPX - Front-End Program

Programming Language: PL/1

This is a front-end program for invoking Natural via EXEC CICS XCTL.

XNCIFRNL - Front-End Program

Programming Language: Assembler

This is a front-end program for invoking Natural via EXEC CICS LINK.

XNCIFRNS - Front-End Program

Programming Language: Assembler

This is a front-end program for invoking Natural via EXEC CICS START.

XNCIFRNP - Initialization Program

Programming Language: Assembler

This program initializes the Natural CICS environment at CICS start-up.

XNCIFRNX - Front-End Program

Programming Language: Assembler

This is a front-end program for invoking Natural via EXEC CICS XCTL.

XNCIGNIT - "Good-Night" Program

Programming Language: Assembler

This "good-night" sample program calls NCIZNEP for Natural session clean-up.

XNCIBACK - Termination Data Dump

Programming Language: Assembler

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 ("STR=xxx" or "RET=xxx"), where "xxx" is the transaction code associated with XNCIBACK.

XNCIRDC1 - Exit for SYSRDC

Programming Language: Assembler

This program provides a sample exit for the SYSRDC utility; see the relevant section in the Natural Utilities documentation.

XNCIUIDX - User Exit

Programming Language: Assembler

This program provides a sample user exit to test/set the user ID (see also NCIUIDEX User ID Exit Interface).

XNCIUCTR - U/L Case Switch

Programming Language: Assembler

This program serves to switch the terminal into upper/lower case mode.

XNCINEP1 - Node Error Program

Programming Language: Assembler

This node error program calls NCIZNEP using the CICS macro level.

XNCINEP2 - Node Error Program

Programming Language: Assembler

This node error program calls NCIZNEP using the CICS command level.

XNCITIDX - User Exit

Programming Language: Assembler

This program provides a sample user exit to test the terminal ID and/or to set a logical terminal or session ID.

Sample Programs for Use with VSE/ESA

For VSE/ESA, the sample programs written in Assembler are supplied as A books. The sample programs written in COBOL are supplied as C books.

Invoking Natural from User Programs

This part of the Natural CICS Interface documentation describes the various ways of how Natural can be invoked from user programs. It covers the following topics:

- Commands for Activating a Natural Session
 - Front-End Parameters
 - Front-End Invoked via LINK
 - Front-End Invoked via START
 - Front-End Invoked via XCTL
 - Invoking Front-End Program as Back-End
-

Commands for Activating a Natural Session

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 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` (Step 15 in the section *Installing the Natural CICS Interface of the Natural Installation Guide for Mainframes*).

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.

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	<p>Invoking transaction ID</p> <p>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.</p> <p>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).</p>
5 - 8	<p>Address/offset of dynamic parameter string</p> <p>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.</p> <p>When being called with START, 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.</p>
9 - 10	<p>Length of the dynamic parameter string</p> <p>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.</p>
11 - 14	<p>Natural transaction ID</p> <p>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.</p> <p>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.</p>
15 - 22	<p>Back-end program name</p> <p>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.</p> <p>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.</p> <p>For further details on the PROGRAM parameter, see Profile Parameters in the Natural Operations for Mainframes documentation. For the conventions of calling non-Natural back-end programs, refer to the Natural Operations for Mainframes documentation.</p>

Front-End Invoked via LINK

If Natural is running in pseudo-conversational mode (PSEUDO=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 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 NCOMxxxx 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 Natural Parameter Reference documentation).

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

In CICSplex Environments

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

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 STARTed), 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.

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.

Asynchronous Natural Processing under CICS

This document contains special considerations that apply when when you are using asynchronous Natural under CICS. The following topics are covered:

- Asynchronous Natural Processing
- Asynchronous Natural Sessions under CICS
- Testing and Debugging

Asynchronous Natural Processing

Asynchronous Natural processing is generally discussed in the section Asynchronous Processing in the Natural Operations for Mainframes 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 Natural parameter module NATPARM and defined in CICS.

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

DUMMY	Any output is ignored.
CONSOLE	Any output is routed to the operator console. Internally the terminal ID defined via the CONSOLE parameter of the NCMPRM macro is used. When dealing with the console, the terminal type should be switched accordingly, using the profile parameter TTYPE or %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.



Attention:

When SET CONTROL 'T=xxxx' or SET CONTROL '+' is used, or PC=ON is specified, the Natural system variable *DEVICE will be modified, which means that it can no longer be used to determine an asynchronous Natural session.

Testing and Debugging

Recent CICS versions offer a transaction CEDX which enables tracing of asynchronous tasks in CICS. In earlier CICS versions, this functionality did not exist, that is, such debugging was only possible with terminal-bound tasks.

The Natural CICS Interface NCI offers some assistance in this case: You can test asynchronous Natural sessions by starting that session from a terminal, but either with "ASYN," as the very first five characters in the dynamic parameter string, or with the profile parameter TTYPE=xxx, where xxx is ASYN or ASYL. The Natural CICS interface then sets up an asynchronous Natural session.

Please, note that this emulation is only 100 percent in terms of Natural; CICS keeps on treating the task as terminal bound.

Logging Natural Sessions under CICS

This section describes how information about Natural sessions can be logged in a file which can be processed and evaluated in batch mode. It covers the following topics:

- Logging Facility
 - Natural Log File Definition
 - Natural Log Records
-

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, swap 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 (if the file runs full in VSE/ESA, the VSE message NO MORE AVAILABLE EXTENTS is issued and the operator is asked to enter new extents or cancel CICS).

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 NCMPRM macro.

The log file has to be defined in a CICS DCT as TYPE=EXTRA with associated dataset control information (TYPE=SDSCI entry in DCT). This file must also be defined in the CICS start-up JCL (DD statement in OS/390, DLBL statement in VSE/ESA).

Natural Log Records

The following records are logged in the Natural log file:

- Natural CICS System Restart Record
- Natural Session Termination Record

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. 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/swap ins/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 SYSTP System Administration Facilities (as described in 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.

Natural CICS Performance Considerations

This section contains guidelines for setting up Natural in a CICS environment. It covers the following topics:

- Environment-Specific Considerations
 - Choosing the Roll Facility
 - Shared Storage Threads versus GETMAINed Threads
 - CICS Parameter Settings
 - Line Compression Systems
 - Pseudo-Conversational versus Conversational Transactions
 - Natural and Adabas
 - CICS Monitoring Products
-

Environment-Specific Considerations

The following environment-specific considerations should be noted:

- When running Natural in a CICSplex environment (OS/390 only), you must use the Natural Roll Server.
- When running Natural locally in a single CICS region, however, you have several possibilities.

One possibility (OS/390 only) is to use the Natural Roll Server. The benefit of this versus using CICS roll facilities and a swap pool is that the Natural Roll Server runs asynchronously to the CICS region and can provide more roll buffers in its data space than the swap pool.

Choosing the Roll Facility

- Control Interval
- VSAM Roll Files versus CICS Temporary Storage
- Using CICS Auxiliary Temporary Storage
- Using CICS Main Temporary Storage
- Using VSAM RRDS Roll Files
- Using the Natural Swap Pool under CICS

Control Interval

You are strongly recommended to define both roll facilities, VSAM and auxiliary temporary storage, with the largest possible control interval size of 32 KB. This minimizes the number of I/Os and the CPU overhead necessary to perform the rolling.

Reasons for a control interval size of less than 32 KB might be the 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 $2n$ 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 CICS Auxiliary Temporary Storage

The primary roll facility is VSAM RRDS; the default type of temporary storage is AUXILIARY.

If you are using VSAM roll files, the Natural CICS interface uses temporary storage (AUX) if all roll files become full or unusable during a CICS session.

However, if you do not wish to use roll files or if the roll files are incorrectly installed, Natural under CICS uses temporary storage (AUX) for all rolling. When temporary storage (AUX) is used as roll file, the control interval size for this file must be at least 4 KB. If auxiliary temporary storage is not available, main temporary storage is used instead.

The number of VSAM buffers defined by the CICS SIT parameter TS should be increased to a reasonable value to reduce the number of physical I/Os. The CICS statistics should be checked for bottlenecks in this area.

Using CICS Main Temporary Storage

With CICS main temporary storage as roll facility, no I/O is performed on rolling, but due to large main storage amounts used, tuning considerations may be required due to increased paging.

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 dataset 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.

Using the Natural Swap Pool under CICS

You are strongly recommended to use a swap pool rather than a large number of VSAM temporary storage (AUX) buffers or temporary storage (MAIN).

The Natural swap manager handles the compressed session storage very efficiently and reduces CPU and I/O overhead. The size of the swap pool should be as large as possible. For example, a swap pool of 2.5 MB would be required to hold 50 sessions which fit into 50 KB slots.

From a performance point of view, it does not make any sense to use main temporary storage as a backup facility for the swap pool, since both of these facilities use CICS main storage. In general though, using the swap pool is more advantageous, because CICS services overhead is eliminated. Rather than overflowing to main temporary storage, it would be better to enlarge the swap pool and to use disk storage (that is, VSAM roll files or auxiliary storage) as its backup facility.

If virtual storage becomes a bottleneck, the number of roll facility buffers and possibly the number of threads should be minimized to the benefit of the swap pool.

When using the Natural swap pool cache, a roll buffer of the size of the largest Natural thread is required for transferring Natural session data between the swap pool and its (data space) cache. This roll buffer is taken from the GETMAIN for the swap pool, that is, the size of the storage actually available for the swap pool is the specified size minus the size of the largest Natural thread.

Therefore a Natural swap pool cache is only allocated when both the size of the swap pool and the size of its cache are at least twice the size of the largest Natural thread.

Shared Storage Threads versus GETMAINED Threads

- Storage Usage
- Controlling Storage Usage
- Swapping/Rolling
- Considerations for CICS 4.1 and above
- Conclusion

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 (TCLASS prior to CICS Version 4.1). 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 (or by using the CMXT parameter of the CICS system initialization table (SIT) prior to CICS Version 4.1).

Swapping/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 or swapping 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/swap 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 4.1 and above

The most important feature of CICS 4.1 and above 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 4.1 and above.

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 NCISTART (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/ESA 4.1, 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/ESA 4.1 and above, because of transaction isolation, the storage of another task may not be accessible at all.

To accomplish this function in CICS/ESA 4.1 and above, NCISTART 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/ESA 4.1 and above, TYPE=GETM threads are preferred, because of:

- the support of transaction isolation,
- 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 the 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/Os 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.



Attention:

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 Natural SYSTP utility should be used instead; see the relevant section in the Natural Utilities documentation for details.

Natural Print and Work Files under CICS

This document discusses the use of Natural print and work files under CICS. It covers the following topics:

- Customizing Print and Work File Usage
 - CICS Temporary Storage Print and Work Files
 - CICS Transient Data Print and Work Files
-

Customizing Print and Work File Usage

The 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 NETWORK macro.
- PRINT profile parameter description and how to set the above subparameter values, see the NTPRINT macro.

The 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 as PRTY, CLASS, COPIES, etc., are not honoured 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 NCIPARM TERMVAR parameter (&TID is the default) in the subparameter DEST of the 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.

System Queues

In Natural under CICS, system queues cannot be accessed. (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/Os against such a file.

Natural under Com-plete/SMARTS

This document describes how to operate Natural in a Com-plete environment.

The following topics are covered:

- Use of the Abend Exits
- Storage Usage
- Support of Back-end Programs
- Com-plete Support in Natural Batch Runs
- Support of Asynchronous Natural Processing
- Invoking Natural from User Programs
- Storage Thread Key Handling
- Support of User Exit Handling during Session Initialization
- Use of the SMARTS Server Environment

See also:

- For further details of the Com-plete product, refer to the Com-plete documentation set.
 - For details concerning the following topics, refer to the Natural Installation Guide for Mainframes:
 - Structure and Functionality of the Natural Com-plete/SMARTS Interface
 - Prerequisites
 - Installation Tape for the Natural Com-plete/SMARTS Interface
 - Installation Procedure for the Natural Com-plete/SMARTS Interface
 - Installation of the Natural Server under Com-plete
 - Using the Com-plete *ULIB Function
 - Installation Verification
 - Customizing a Natural Com-plete Environment
 - The Natural utility SYSTP provides various TP-monitor-specific functions (see SYSTP Utility).
 - See also Natural under Com-plete Abend Codes in the Natural Codes and Messages documentation.
-

Use of the Abend Exits

The ABEXIT exits can generally be deactivated by setting SPIEA=NO in NCFPARM.

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

By default, an 0CX abend is interpreted by the ABEXIT exit routine.

- Running with DU=ON/SNAP/ABEND, the Natural session is dumped and correctly terminated with error message NAT9974.
- Running with DU=FORCE, the ABEXIT exit routine is disabled, an immediate dump during Com-plete is produced.

If DU=OFF, Natural responds with error message 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 parameter NTHSIZE in NCFPARM is allocated as thread GETMAIN above or below the 16 MB line, depending on the parameter THABOVE, for usage by Natural.

The WPSIZE profile parameter 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 things:

- 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 of 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.

Support of Asynchronous Natural Processing

Asynchronous Natural processing is discussed in the section Asynchronous Processing in the Natural Operations for Mainframes documentation; however, some additional considerations apply when running Natural 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.

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 profile parameter SKEY=OFF in the Natural parameter module NATPARM. The application program runs in the corresponding thread key. For any Natural or Editor buffer pool call, the front-end driver switches into the appropriate key and back to the thread key after the call.

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

Set the thread key = 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 using the Com-plete function MODIFY with function codes THRD/TCS.

Support of 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 can be defined in the parameter module NCFPARAM.

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 datasets 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 datasets are accessed as described in the section *Datasets Used by Natural under OS/390 Batch* in the *Natural Operations for Mainframes* documentation.

The following topics are covered below:

- Input/Output
- Print File/Work File

Input/Output

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

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

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 dataset, 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 dataset *pathname/filename* will be accessed; if it is not terminated with "/", the member *filename* in dataset *pathname* will be accessed.

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 dataset natural.test is accessed.

CMPRINT = /fs/natural/test/print/ Sequential dataset 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 dataset *pathname/filename* will be accessed; if it is not terminated with "/", the member *filename* in dataset *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 "/" character, the sequential dataset *pathname/filename* will be accessed; if it is not terminated with "/", the member *filename* in dataset *pathname* will be accessed.

Print File/Work File

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

NCFWFAPS must be loaded into the resident area. If NCFWFAPS is loaded into the application program thread, the Natural session is terminated with NAT9980.

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 system.

Natural under IMS/TM - Overview

This document describes how to operate Natural in an IMS/TM environment.

The following topics are covered:

- Environments
- Components
- Configuration Macros
- Service Programs
- Service Modules
- User Exits
- Special Functions
- Recovery Handling

See also:

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

Natural under IMS/TM - Environments

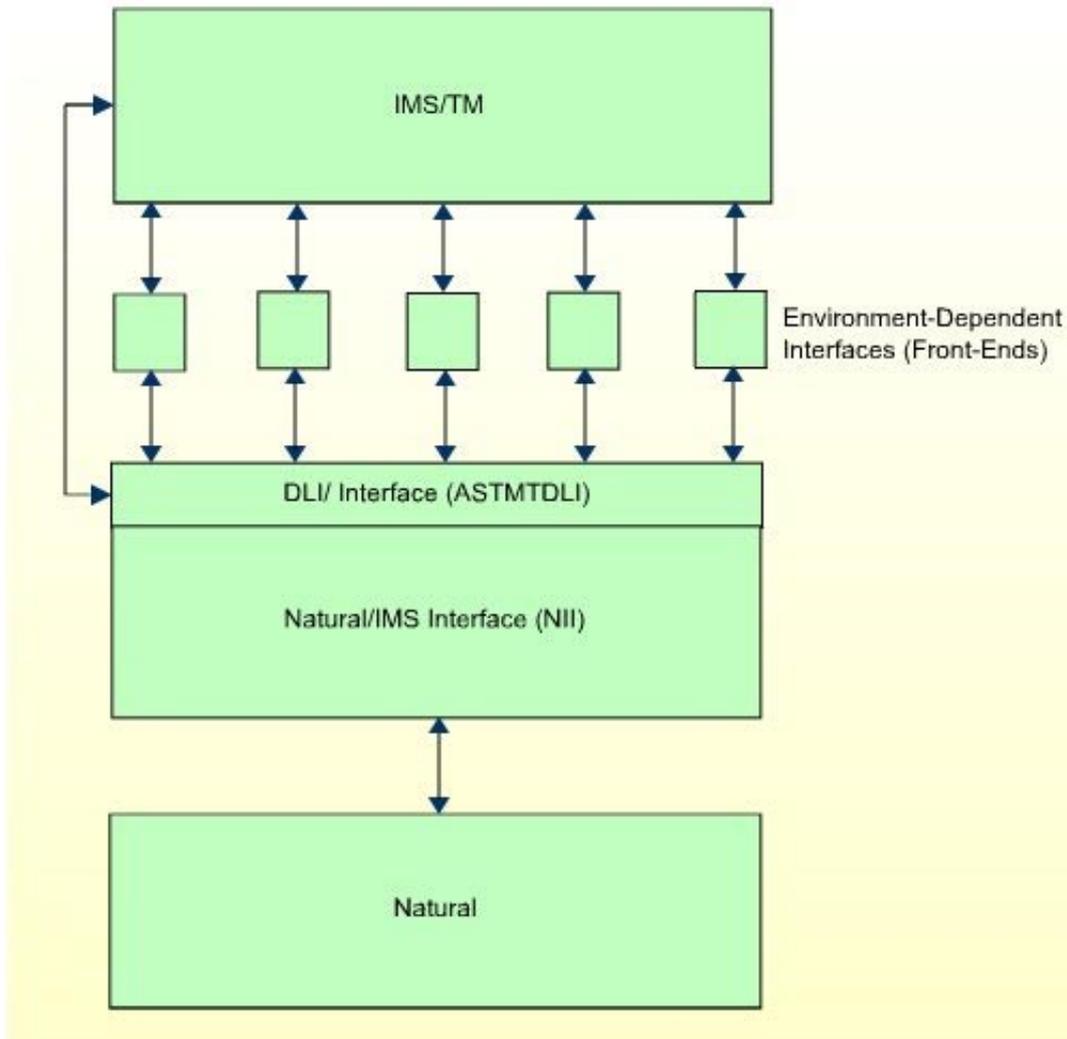
This document describes how Natural runs under various IMS/TM environments. The following topics are covered:

- IMS/TM Interface Overview
 - IMS/TM Environments
 - Dialog-Oriented Environments
 - Message-Oriented Environment
 - Batch Message Processing Environment
 - Support of the Natural WRITE (*n*) Statement
 - SENDER Destination
 - Support of Natural Profile Parameter PROGRAM
-

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes (in the Natural Messages and Codes documentation).
 - **Installation** - refer to Installing the Natural IMS Interface (in the Natural Installation Guide for Mainframes).
 - **Further information** - refer to the following topics:
Components | Configuration Macros | Service Programs | Service Modules | Special Functions | User Exits | Recovery Handling
-

IMS/TM Interface Overview



IMS/TM Environments

IMS/TM provides three different types of environments:

- message processing regions,
- batch message processing regions,
- off-line DL/I btach region.

To be able to use Natural in each of these environments, different environment-specific interfaces are provided for the Natural IMS 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 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,
- message-oriented.

Dialog-Oriented Natural

A dialog-oriented Natural session establishes an ongoing interaction with an IMS 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 interface detects an error situation, a record containing information about this error situation is written to the IMS 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 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 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.

Natural in an Off-line DL/I Batch Region

The BMP Natural can also be executed as an off-line DL/I batch job.

If no IOPCB is available, all END TRANSACTION and BACKOUT TRANSACTION statements are ignored.

For diagnostic purposes, the following feature is available: If Natural has been started with dynamic profile parameter TPF=(1), an informal WTO message is issued, indicating the above fact.

Dialog-Oriented Environments

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

- Special Considerations for a Conversational Environment
- Special Considerations for a Non-Conversational Environment
- Special Considerations for an MSC Environment

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 NRASTART value.

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 Authorized Services Manager with its SIP function enabled and the Physical Input Edit Routine are prerequisites.

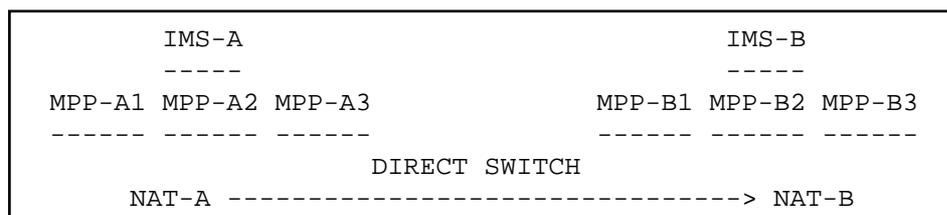
- The Authorized Services Manager is used to simulate the IMS/TM SPA.
- The Physical Input Edit Routine is used to insert the transaction code in front of the input message.

You must specify the same Natural subsystem ID in the

- SPATID of the NIMPARM macro,
- 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 interface prepares the message X'000500006D' for NAT-B, which means that the terminal user has pressed CLEAR.



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
- Operation of the Message-Oriented Environment

Introduction to the Message-Oriented Environment

This interface is designed to process net-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 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 DL/I and Natural for DB2 process 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 and all DL/I requests within this Natural session are considered to be related, requiring maintenance of database positioning and PCB usage. With the non-conversational interface, this Natural for DL/I logic is not applicable.

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 dataset 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 message queue is accessed by your application prior to the termination of Natural. If not, the Natural transaction abnormally ends with IMS 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 dataset 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 above.

Example:

```
ENV-TAB=ENVBMP0 TRNCODE=NATIMS
```

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

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

If the CMPRMIN dataset 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 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 parameter WRKPCBs.



Attention: Be aware that the first alternate TP-PCB is used by the Natural IMS 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 parameters of the NTPRINT macro are relevant:

AM | DEST | BLKSIZE | DRIVER | NAME | FORMS | DISP | COPIES | CLASS | PRTY

Parameter	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 parameter of the Natural/IMS interface 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 NII 2.2 compatibility) 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/PRINT definition for IMS/TM printers (i.e. for printers defined with AM=STD). This means either don't specify any values for OPEN and CLOSE or use the defaults OPEN=ACC and CLOSE=CMD.

This is especially important if you have statically defined a printer in the NATPARM 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 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.

Usage of CLOSE PRINTER or DEFINE PRINTER

A report written to an IMS/TM printer is implicitly closed by IMS/TM with the next GU call (i.e. 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:



A CLOSE PRINTER 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 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

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 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 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 (e.g. 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 parameter 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 (e.g. 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 OS/390 batch environment. An example with name XNIIBACK is delivered in the NIIvrs.SRCE dataset. It is expected that the invoked back-end program returns to the Natural IMS 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 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 transaction.

Natural under IMS/TM - Components

This part of the Natural IMS Interface documentation discusses the components of the Natural IMS interface. The following topics are covered:

- Front-End Module
- Natural IMS Interface Module NIIINTFM
- Physical Input Edit Routine
- Roll File and Roll Server
- Shared Natural Nucleus
- Natural Buffer Pool
- Adabas Interface
- Preload List

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
- **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
- **Further information** - refer to the following topics:
Environments | Configuration Macros | Service Programs | Service Modules | Special Functions | User Exits | Recovery Handling

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 created during the installation process and consists of the following:

- Environment-dependent Interfaces (Drivers)
- Natural Parameter Module NATPARAM
- Natural Work and Print File Access Method Module NATWKFO (AM=STD)
- Modules from Other Natural Products

Environment-Dependent Interfaces (Drivers)

You must generate an environment-dependent interface for each IMS environment supported by the Natural IMS interface using the NIMDRIV macro.

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

Natural Parameter Module NATPARAM

For information on the Natural parameter module, see Assembling a Natural Parameter Module (in the Natural Operations for Mainframes 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 defined with AM=STD. It is applicable to the BMP environment, including off-line DL/I 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 and Natural for DL/I, require that their modules be linked to the Natural IMS front-end module. For further information, see the appropriate product documentation.

Natural IMS Interface Module NIINTFM

The Natural IMS 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 Nucleus
- Natural IMS Parameter Module NIIPARM
- Transaction Code Table NIITRTAB
- Message Text Module NIIMSGT
- DL/I Language Interface ASMTDLI

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

Natural IMS Nucleus

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

Natural IMS Parameter Module NIIPARM

The Natural IMS parameter module NIIPARM contains a number (1 - *n*) of parameter tables (also called "environment tables") each defined by the macro NIPARM and identified by the parameter ENTRYNM. Each parameter table within NIIPARM defines the Natural IMS-specific parameters for a particular environment. Thus it is possible to set the parameters for all Natural IMS environments in one parameter module. The environment which is currently used is set in the transaction code table NIITRTAB.

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

Transaction Code Table NIITRTAB

The transaction code table NIITRTAB table is of variable length and each entry is generated by the macro NIMTRNTG.

Each entry in the transaction code table refers to an entry in the Natural IMS parameter module and can be followed by one or more occurrences of the macro NIMLPCB.

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

The appropriate entry within this table is detected by the current transaction code. If, in a non-message-driven BMP, no transaction code is defined, the current PSB name is taken instead.

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

Message Text Module NIIMSGT

The message text module NIIMSGT is part of the Natural IMS Interface module and is supplied both as a load and a source module. For each possible Natural IMS 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 .

DL/I Language Interface ASMTDLI

The DL/I language interface ASMTDLI is part of IMS/TM.

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.

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 NIMPARM parameter `ROLLSRV`.

Using Roll Files

To use roll files, the parameter `ROLLSRV` is set to `NO`.

A roll slot in the roll file is reserved for each Natural user at Natural session initialization time. 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 NIMPARM parameter `ROLLFN`.

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 after the roll file has been reformatted.

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

Using the Roll Server

To use the roll server, the parameter `ROLLSRV` is set to `YES`.

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.
- Data space support.

In a 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 for Mainframes documentation).

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 environments.

For further information, see Natural Shared Nucleus (in the Natural Operations for Mainframes documentation).

Natural Buffer Pool

Since Natural under IMS 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 for Mainframes 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 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.



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

Natural under IMS/TM - Configuration Macros

This part of the Natural IMS Interface documentation discusses the configuration macros of the Natural IMS interface.

The following topics are covered:

- NIMDRIV Macro Parameters
- NIMPARM Macro Parameters
- NIMTRNTG Macro Parameters
- NIMLPCB Macro Parameters
- NIMMSGT Macro Parameters
- NIMPIXT Macro Parameters
- NIMBOOT Macro Parameters

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
 - **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
 - **Further information** - refer to the following topics:
Environments | Components | Service Programs | Service Modules | Special Functions | User Exits | Recovery Handling
-

NIMDRIV Macro Parameters

The macro NIMDRIV generates environment-dependent interfaces (drivers). The parameters which can be specified with the macro NIMDRIV are described below:

TYPE | LE370 | NIINAME | SUBPOOL | TRNCODE | THRELO

Parameter	Possible Values	Description	Default	Comment
TYPE	CONV	TYPE specifies the type of the driver to be generated. In case of "CONV", a dialog-oriented conversational environment is generated.	None	None.
	NONC	Dialog-oriented non-conversational environment is generated.		
	NTRD	Message-oriented (not terminal-driven) environment is generated.		
	BMP	Batch-oriented and transaction-oriented BMP environment is generated. Note: You can also use this interface in the DLIBATCH environment.		
	SRVD	Server environment is generated.		
LE370	YES	LE370 specifies whether Natural under IMS initializes the IBM Language Environment. In case of "YES", the IBM Language Environment is initialized and remains so until the Natural IMS front-end returns to the IMS program controller. LE-conforming 3GL programs are called according to the LE calling conventions. While the LE program has control, user written or language-specific condition handlers are honored.	NO	None.
	NO	The IBM Language Environment is not initialized. An IBM Language Environment is initialized and terminated for each LE program call.		
NIINAME	xxxxxxx	Specifies the name of the Natural IMS interface module to be used by the current driver. Any valid module name up to 8 characters is possible..	NIINTFM	None.
SUBPOOL	<i>n</i>	Any valid OS/390 subpool number for a problem state program (0 Æ 127, 131 and 132).	127	
TRNCODE	YES	The optional TRNCODE parameter is honored. This applies to the <ul style="list-style-type: none"> • TRNCODE parameter on the first card of the BPM CONTROL file, • TRNCODE parameter of the NIMBOOT macro, • TRNCODE startup parameter of the server environment. 	NO	
	NO	The TRNCODE parameter is ignored.		

THRELO	YES	Thread relocation is enabled, that is, the Natural thread may be allocated at a different virtual address after a terminal I/O.	YES	
	NO	<p>Thread relocation is disabled and the Natural thread stays at the same virtual address. The size of the Natural thread is determined by the first Natural session that allocates the thread. As consequence:</p> <ul style="list-style-type: none"> ● the Natural/IMS THSIZE parameter is ignored by all except for the first Natural session in an MPP, ● RELO=FORCE is disabled, ● the storage for the Natural thread remains allocated until the MPP is stopped. <p>Note: This parameter should only be used by customers using Con-form.</p>		

NIMPARM Macro Parameters

The macro NIMPARM generates parameter tables which are contained in the parameter module NIIPARM.

The parameters which can be specified with the NIMPARM macro are described below. The ENTRYNM parameter identifies the current parameter table.

The parameters are listed in alphabetical order below.

ACTACTV | ACTAHDR | ACTARID | ACTLOG | BMPABER | BROACTV | CMBSIZE | COLPSCR | ENDMODN | ENTRYNM | ERR LHDR | HCBSIZE | HDENS DU | LINPSCR | MISIZE | MONACTV | MOSIZE | MSACTV | MSCMPTB | MSCRKEY | MSDBD | MSMAX | MSRSKEY | NSBNAME | PRTDRIV | ROLLSRV | ROLLFN | SPASIZE | SPATID | SUPNONC | TERMDB | TERMIPL | THBELOW | THSIZE | USERID

A

Parameter	Possible Values	Description	Default	Comment	
ACTACTV	YES	ACTACTV specifies whether the accounting function is activated. In case of "YES", an accounting record is written with each terminal I/O.	NO	Used in dialog-oriented environments only.	
	NO	No accounting record is written.			
ACTAHDR	xxxxxxx	Defines the header of the accounting records if written to the IMS log file. Any string up to 8 bytes is possible.	SAG\$\$\$\$\$	The parameter is only evaluated when the ACTLOG parameter is set to CMD. It is used for the accounting function only.	
ACTARID	<i>log code</i> or <i>SMF record type</i>	Specifies the accounting record ID if the accounting record is written using the LOG or SMF settings of the ACTLOG parameter.	None	Used for the accounting function only.	
		Log code (A0 - FF)			When the ACTLOG parameter (see below) is set to LOG.
		SMF record type (128 - 255)			When the ACTLOG parameter is set to SMF.
ACTLOG	CMD	ACTLOG specifies how accounting records are written. In case of "CMD", accounting records are written to the IMS log file using the CMD call.	CMD	Used for the accounting function only.	
	LOG	Accounting records are written to the IMS log file using the LOG call.			
	SMF	Accounting records are written to SMF using Authorized Services Manager.			

B - C

Parameter	Possible Values	Description	Default	Comment
BMPABER	YES	BMPABER specifies how a BMP run should be terminated if either a Natural runtime error or a Natural IMS interface non-recoverable error occurs. In case of "YES", the BMP run is abended with ABEND code 3521.	NO	None.
	NO	The BMP run is terminated normally with the Natural termination error as the condition code. If the BMP run is terminated with a non-recoverable Natural IMS error, condition code 1024 is set.		
BROACTV	YES	Specifies whether the broadcasting function is available or not.	NO	Used in dialog-oriented environments only.
	NO			
CMBSIZE	xxxxx	Specifies the size of the command buffer in bytes. Any numeric value up to 16 MB is possible.	1024	The command buffer is used by the service APIs NIICMD and NIIGCMD, the service module CMCMMND and the Accounting function. The size of the command buffer must accommodate the maximum length of the IMS commands to be processed and the maximum length of the accounting record including the user extension.
COLPSCR	xx	Specifies the number of columns per screen. Any valid screen width (numeric) is possible.	80	None.

E - H

Parameter	Possible Values	Description	Default	Comment
ENDMODN	xxxxxxx	Specifies the MOD name for formatting the screen which appears after a Natural session is terminated successfully. Any valid MOD name up to 8 characters is possible.	DFSMO2	<p>Enables Natural to be included in a customer-specific menu.</p> <p>The value of the ENDMODN parameter can be overridden by the service API NIIEMOD and the service module CMEMOD.</p> <p>If a Natural session terminates with an error, DFSMO2 is always used to issue the appropriate Natural error message.</p>
ENTRYNM	xxxxxxx	Identifies the current parameter table. Any string up to 8 characters is possible.	ENV00000	None.
ERRLHDR	xxxxxxx	Specifies the header of the IMS log records which are written when errors occur in the Natural IMS interface. Any string up to 8 characters is possible.	NIIERR\$\$	<p>If you do not wish a message to be written to the IMS log in the case of a non-recoverable Natural IMS error, set the ERRLHDR parameter explicitly to null, i.e. specify "ERRLHDR=,".</p> <p>For further information, see Recovery Handling.</p>
HCBSIZE	xxxxx	Specifies the size in bytes of the hardcopy print buffer. Any numeric value up to 16 MB possible.	1024	Records which are sent to a printer destination using the Natural hardcopy function are buffered.
HDENSUDU	YES NO	Specifies whether a snap dump provoked by a Natural IMS error should be written as a high-density dump to a 3800 printing subsystem or not.	NO	None.

L - N

Parameter	Possible Values	Description	Default	Comment
LINPSCR	<i>xx</i>	Defines the number of lines per screen. Any valid screen size (numeric) is possible.	24	None.
MISIZE	<i>xxxxx</i>	The size in bytes of the buffer which is to contain the input message. Any numeric value up to 16 MB is possible.	4096	This area must be as large as the largest input message to be received from IMS/TM.
MONACTV	YES	MONACTV specifies whether the monitoring function is activated. In case of "YES", the session status is written to the SIP server at each terminal I/O.	NO	Used in dialog-oriented environments only.
	NO	No session status is maintained.		
MOSIZE	<i>xxxxx</i>	The size in bytes of the buffer which is to contain the output message. Any numeric value up to 16 MB is possible.	4096	This area must be as large as the largest output message to be sent to IMS/TM.
MSACTV	YES	Specifies whether the multi-session function is available or not.	NO	Used in dialog-oriented environments only.
	NO			
MSCMPTB	YES	Specifies whether sessions are switched in NII22/NIA-compatible mode or not.	NO	Used for the multi-session function only.
	NO			
MSCRKEY	NONE	Specifies with which PF keys a new session can be started.	NONE	Used for the multi-session function only. If MSCMPTB=YES, MSCRKEY must be set to NONE.
	PF1 - PF24			
MSDBD	<i>DBD name</i>	Specifies the name of the multi-session database. Any valid DBD name is possible.	None	Used for the multi-session function only.
MSMAX	2 - 9	Specifies the highest possible number of parallel Natural sessions per terminal.	9	Used for the multi-session function only.
MSRSKEY	NONE	Specifies the PF key with which an old session can be restarted.	NONE	Used for the multi-session function only. MSRSKEY must be set to NONE, if MSCMPTB=YES.
	PF1 - PF24			
NSBNAME	PSBNAME	Set the NSB name to the PSB name used by IMS/TM.	PSBNAME	Used by the Natural for DL/I interface.
	NIITRTAB	Set the NSB name to the PSB name that is assigned in the transaction code table NIITRTAB to the transaction code in use.		

P

Parameter	Possible Values	Description	Default	Comment
PRTDRIV	See tables of drivers below. Drivers for SCS Printers Drivers for Non-SCS Printers Drivers for JES API	Specifies the print driver to be used for reports which are directly written to an IMS/TM printer.	SCS_S2	For further information, see Support of the Natural WRITE (<i>n</i>) Statement.

Drivers for SCS Printers

Driver	Purpose
SCS_B1	Form feed at start and end of report, starts page on line 1.
SCS_B2	Form feed at start and end of report, starts page on line 2.
SCS_E1	Form feed at end of report, starts page on line 1.
SCS_E2	Form feed at end of report, starts page on line 2.
SCS_N1	No form feed at start or end of report, starts page on line 1.
SCS_N2	No form feed at start or end of report, starts page on line 2.
SCS_S1	Form feed at start report, starts page on line 1.
SCS_S2	Form feed at start of report, starts page on line 2.

Drivers for Non-SCS Printers

Driver	Purpose
NSCS_B1	Form feed at start and end of report, starts page on line 1.
NSCS_B2	Form feed at start and end of report, starts page on line 2.
NSCS_E1	Form feed at end of report, starts page on line 1.
NSCS_E2	Form feed at end of report, starts page on line 2.
NSCS_N1	No form feed at start or end of report, starts page on line 1.
NSCS_N2	No form feed at start or end of report, starts page on line 2.
NSCS_S1	Form feed at start report, starts page on line 1.
NSCS_S2	Form feed at start of report, starts page on line 2.

Drivers for JES API

Driver	Purpose																								
JES	In this case, the following dataset processing options for JES are taken from the corresponding NTPRINT or DEFINE PRINTER parameters:																								
	<table border="1"> <thead> <tr> <th data-bbox="312 497 655 539">JES</th> <th data-bbox="655 497 999 539">NTPRINT</th> <th data-bbox="999 497 1398 539">DEFINE PRINTER</th> </tr> </thead> <tbody> <tr> <td data-bbox="312 539 655 584">CLASS</td> <td data-bbox="655 539 999 584">CLASS</td> <td data-bbox="999 539 1398 584">CLASS</td> </tr> <tr> <td data-bbox="312 584 655 629">COPIES</td> <td data-bbox="655 584 999 629">COPIES</td> <td data-bbox="999 584 1398 629">COPIES</td> </tr> <tr> <td data-bbox="312 629 655 674">DEST</td> <td data-bbox="655 629 999 674">DEST</td> <td data-bbox="999 629 1398 674">OUTPUT</td> </tr> <tr> <td data-bbox="312 674 655 719">FORMS</td> <td data-bbox="655 674 999 719">FORMS</td> <td data-bbox="999 674 1398 719">FORMS</td> </tr> <tr> <td data-bbox="312 719 655 763">NAME</td> <td data-bbox="655 719 999 763">NAME</td> <td data-bbox="999 719 1398 763">NAME</td> </tr> <tr> <td data-bbox="312 763 655 808">OUTDISP</td> <td data-bbox="655 763 999 808">DISP</td> <td data-bbox="999 763 1398 808">DISP</td> </tr> <tr> <td data-bbox="312 808 655 853">PRTY</td> <td data-bbox="655 808 999 853">PRTY</td> <td data-bbox="999 808 1398 853">PRTY</td> </tr> </tbody> </table>	JES	NTPRINT	DEFINE PRINTER	CLASS	CLASS	CLASS	COPIES	COPIES	COPIES	DEST	DEST	OUTPUT	FORMS	FORMS	FORMS	NAME	NAME	NAME	OUTDISP	DISP	DISP	PRTY	PRTY	PRTY
	JES	NTPRINT	DEFINE PRINTER																						
	CLASS	CLASS	CLASS																						
	COPIES	COPIES	COPIES																						
	DEST	DEST	OUTPUT																						
	FORMS	FORMS	FORMS																						
	NAME	NAME	NAME																						
	OUTDISP	DISP	DISP																						
	PRTY	PRTY	PRTY																						
<p>The generated JES API parameter string is:</p> <pre>IAFP=A0M,PRTO=..OUTDI(<i>disp</i>),DES(<i>dest</i>),CLA(<i>class</i>),COP(<i>copies</i>), FORMS(<i>forms</i>),NAME(<i>name</i>),PRTY(<i>prty</i>)</pre> <p>Note: Unspecified NTPRINT/DEFINE PRINTER parameters are ignored.</p>																									
JESxxxxx	<p>In this case, the dataset processing options for JES are taken from the OUTPUT JCL statement with the name JESxxxxx.</p> <p>The generated JES API parameter string is: IAFP=A0M, OUTN=JESxxxxx</p> <p>The OUTPUT JCL statement may look like:</p> <pre>JESxxxxx OUTPUT OUTDISP=WRITE,DEST=<i>dest</i>,CLASS=A,COPIES=1,FORMS=<i>form</i>,...</pre> <p>Note: If the OUTPUT JCL statement is missing in the job stream, an error is reported.</p>																								

R - S

Parameter	Possible Values	Description	Default	Comment
ROLLSRV	YES	ROLLSRV specifies the medium for saving a Natural thread between terminal output and input. If ROLLSRV=YES, the Natural roll server is used.	YES	Used in dialog-oriented environments only.
	NO	Roll files are used, see ROLLFN below.		
ROLLFN	1 - 5	Specifies the number of roll files to be used, if ROLLSRV=NO.	1	Used in dialog-oriented environments only.
SPASIZE	xxxxx	Specifies the size in bytes of the buffer which is to contain the scratch-pad area. Any numeric value up to 16 MB is possible.	1024	In a non-conversational environment, this is also the size of the simulated SPA which is written to the SIP server.
SPATID	xxxx	Specifies the Natural subsystem ID for the Authorized Services Manager which is used to save the SPA for a non-conversational driver. Any string up to 4 characters is possible.		This value must be the same for all parameter tables and must be the same as the value specified for SPATID in the NIMPIXT macro.
SUPNONC	YES	Specifies whether switching from a terminal-oriented non-conversational environment to a conversational environment is possible.	NO	Used in the dialog-oriented conversational environment only.
	NO			

T - U

Parameter	Possible Values	Description	Default	Comment
TERMDB	YES	Specifies whether the Natural session has to be terminated if one of the DL/I databases specified in the PSB is not available.	NO	Used in dialog-oriented environments only. If you set TERMDB to "NO" and one of the databases is not available when it is accessed, the Natural transaction code is suspended by IMS/TM.
	NO			
TERMIPL	YES	Specifies whether a Natural session is terminated with an error message when an IPL has taken place between the current transaction step and the start of the session.	NO	Used in dialog-oriented environments only.
	NO			
THBELOW	YES	Specifies where the Natural thread is allocated. In case of YES, the Natural thread is allocated below the 16 MB line.	YES	For batch message processing, the thread is always allocated below the 16 MB line.
	NO	The Natural thread is allocated above the 16 MB line.		
THSIZE		Specifies the size of the Natural thread. Any numeric value in multiples of eight greater than or equal to 100000 is possible.	300000	This is the area which contains all user session related Natural buffers.
USERID	YES	USERID specifies how the value of the system variable *init-user is determined. In case of "YES", the Natural user ID specified in *INIT-USER is either taken from the security access control block if a security package is active or from the USER parameter of the job card.	NO	Used by the BMP driver only.
	NO	The Natural user ID specified in *INIT-USER is taken from the job name.		

NIMTRNTG Macro Parameters

The macro NIMTRNTG generates an entry in the transaction code table NIITRTAB containing the specified transaction code with related parameters. For each Natural transaction code an entry has to be included in the transaction code table. For further information on NIITRTAB, see Transaction Code Table NIITRTAB.

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

ALTPCB | HCPCB | MSGPCB | MSPCB | NIIPENT | NRASTART | PSBNAME | TRANCODE | TYPE | WRKPCBS

Parameter	Possible Values	Description	Default	Comment
ALTPCB	1 - 255	ALTPCB specifies which alternate TP PCB is used for the service modules CMQUEUE, CMQUEUEEX, NIIDQUMS and NIIDPURG.	1	The number of the alternate TP PCB specified with the NIMTRNTG macro can be overwritten by the service modules.
HCPCB	SYSPCB	HCPCB specifies which PCB is used for the hardcopy function. In case of "SYSPCB", the first alternative TP PCB is used.	SYSPCB	None.
	WRKPCB	One of the additional alternative TP PCBs is used. This enables you to use an express TP PCB for the hardcopy function.		
MSGPCB	SYSPCB	MSGPCB specifies which PCB is used when printing error messages and standard output in the non-terminal-oriented environment and for the server driver. In case of "SYSPCB", the first alternative TP PCB is used.	SYSPCB	Relevant for non-terminal-oriented environments and the server driver only.
	OWNPCB	The second alternative TP PCB is reserved and used. This enables you to use an express TP PCB for sending messages.		
MSPCB	NO	MSPCB specifies the number of the multi-session database PCB. If "NO" is specified, the multi-session feature is not used.	NO	Relevant for the multi-session feature only.
	1 - 255	The PCB of the multi-session database.		
NIIPENT	xxxxxxx	Specifies the name of the Natural IMS parameter table to be used for this entry in the transaction code table. Any non-blank character string up to 8 characters is possible.	ENV00000	None.

NRASTART	<i>offset value</i>	Defines the offset of the Natural Reserved Area (NRA) within the scratch-pad area. Any numeric value greater than 14 is possible.	16	The current length of the NRA is 157 bytes. The length of the NRA may change from version to version of the Natural IMS interface. If you want to save your own information in the SPA in order to pass it to a non-Natural transaction, it is recommended that you save your data in front of the NRA in order to be version compatible.
PSBNAME	<i>PSB name</i>	Specifies the PSB name corresponding to the current transaction code. Any valid PSB name is possible.	None	Used to identify the entry in the transaction code table for non-message-driven batch message processing and for the batch processing environment.
TRANCODE	<i>transaction code name</i>	Specifies the identifier of each entry within the transaction code table. Any valid transaction code name is possible.	None	Has no effect in both the non-message-driven BMP and the batch processing environment.
TYPE	CONV	TYPE specifies the type of the Natural transaction code. In case of "CONV", the transaction code is for a conversational Natural session.	CONV	None.
	NONC	The transaction code is for a non-conversational Natural session.		
WRKPCBS	0	WRKPCBS specifies the number of alternative TP PCBs available for printing additional to the first TP PCB and, if appropriate, to the MSGPCB. In case of "0", no IMS printer is available.	0	See examples below.
	1 - 32	The number of alternate TP PCBs used for printing additional to the first TP PCB and, if appropriate, to the MSGPCB.		

Examples

Example 1:

You specified the following:

MSGPCB=SYSPCB
WRKPCBS=2

The PSB must contain 3 alternate TP PCBs.

Example 2:

You specified the following:

```
MSGPCB=OWNPCB  
WRKPCBS=2
```

The PSB must contain 4 alternate TP PCBs. The second alternate TP PCB is reserved for the error messages and standard output of the non-terminal-oriented environment.

NIMLPCB Macro Parameters

The macro NIMLPCB can optionally follow a NIMTRNTG entry in the transaction code table.

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

NAME | NUM

Parameter	Possible Values	Description	Default	Comment
NAME	xxxxxxx	Specifies the logical name of the PCB. Any non-blank string up to 8 characters is possible.	None	None.
NUM	<i>PCB positional number</i>	Specifies the positional number of the PCB in the PSB. Any integer is possible.	None	If NUM is not specified, the positional number of the NIMLPCB macro is used.

NIMMSGT Macro Parameters

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

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

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

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

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

In this case, Natural under IMS 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:

NIA | PIXTE | SIPSE | SPATID | SVC | SVCE | WTO | USER

Parameter	Possible Values	Description	Default	Comment
NIA	YES	Specifies whether NIA (Natural under IMS/TM Advanced Interface) is supported by the physical input edit routine.	NO	If you want to run the Natural IMS Interface Version 2.3 and NIA 2.2 in parallel (on the same terminals), specify YES.
	NO			
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.
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.
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 parameter in the NIMPARM macro.
SVC	200 - 255	Specifies the SVC numbers used by NIA.	None	For more information, see the Natural under IMS/TM Advanced Interface documentation (Manual order no. NIA-225-110).
SVCE	1 - 999	Specifies the start value for error numbers if errors are detected by the NIA SVC.	200	This value is added to the return code of the NIA SVC to create the error message number.
WTO	YES	Specifies whether a WTO message is issued if the Authorized Services Manager fails.	NO	None.
	NO			
USER	xxxxxxxx	Specifies whether a user-specific physical input edit routine is to be called if the NIMPIXT macro does not find the SPA.	NO	None.
	NO	If a user-specific input edit routine is to be called, specify the name of the routine.		

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 OS/390 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 (NIMDRIV parameter TYPE=SRVD). If no TYPE is specified the front-end module must have been generated for the message-oriented environment (NIMDRIV parameter TYPE=NTRD).
ENVTNAM	Any valid OS/390 module name	None	This parameter is only used by the bootstrap module for the message-oriented 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 interface. This parameter is optional and is only honoured if TRNCODE=YES is coded on the NIMDRIV macro. If it is not specified or if TRNCODE=YES is not specified on the NIMDRIV 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 NIITRTAB.
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.
SERVERN	Any valid OS/390 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 of them.. See Special Functions, Server Environment.

Natural under IMS/TM - Service Programs

This part of the Natural IMS Interface documentation describes the service programs of the Natural IMS/TM Interface. The following topics are covered:

- Introduction to the Natural IMS/TM Interface Service Programs
 - Description of the Natural IMS/TM Interface Service Programs
-

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
 - **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
 - **Further information** - refer to the following topics:
Environments | Components | Configuration Macros | Service Modules | Special Functions | User Exits | Recovery Handling
-

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 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 error message.

Description of the Natural IMS/TM Interface Service Programs

The following service programs are described below:

NIIBRCST | NIICMD | NIIDEFT | NIIDEFTX | NIIDIRT | NIIDIRTX | NIEMOD | NIIGCMD | NIIGMSG | NIIGSEG | NIIGSPA | NIIMSIN | NIISRTF | NIISRTRM | NIIPCBAD | NIIPCOM | NIIPMSG | NIIPSBAD | NIIPSPA | NIIPURG | NIIRETRM | NIISASD | NIU3962

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:

Terminal_name | Message | Message_length | MOD_name | Return_code

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 Command to IMS

Passes the IMS command specified to IMS. 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:

Command | Command_length | Reply_area | Reply_area_length | Reply_length | Status_code | Return_code

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:

Transaction_code | Return_code

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:

Transaction_code | Transaction_type | Suspend_flag | MOD_name | Message | Message_length | Return_code

Transaction_code	(A8)	Input
Transaction_type	(A4)	Input Possible values: CONV for conversational NONC for non conversational
Suspend_flag	(A1)	Input Possible values: 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.

Transaction_code | Return_code

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:

Transaction_code | Transaction_type | Suspend_flag | Message | Message_length | Return_code

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

NIEMOD - 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:

MOD_name | Return_code

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:

Reply_area | Reply_area_length | Reply_length | Status_code | Return_code

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:

Message_area | Message_area_length | Return_code

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:

Message_area | Message_area_length | Return_code

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:

Offset | Length | Area | Return_code

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 Environment Info

Retrieves the IMS 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:

Reply_area | Reply_area_length | Return_code

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 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:

Destination | Message | Message_length | Return_code

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:

Message | Message_length | Return_code

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:

PSB_name | Logical_PCB_name | PCB_address | Return_code

PSB_name	(A8)	Output
Logical_PCB_name	(A8)	Input
PCB_address	(B4)	Output
Return_code	(I4)	Output

Specific Return Code Values: None.

Sample Program: NIIPCBAD

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:

Offset | Data_area | Length | Return_code

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:

Message | Message_length | MOD_name | Return_code

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:

PSB_address | Return_code

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:

Offset | Length | Data_area | Return_code

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:

Return_code

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:

Offset | Length | Message_area | Return_code

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 dynamic parameters SENDER and OUTDEST.

The following parameters are provided:

Sender | Outdest | Return_code

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 3962 and produces a dump.

The following parameter is provided:

Return_code

Return_code	(I4)	Output
-------------	------	--------

Specific Return Code Values: None.

Sample Program: NIPU3962

Natural under IMS/TM - Service Modules

This document describes the service modules of the Natural IMS interface. The following topics are covered:

- Purpose of Service Modules
- Service Module Descriptions

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
- **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
- **Further information** - refer to the following topics:
Environments | Components | Configuration Macros | Service Programs | Special Functions | User Exits | Recovery Handling

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 Operator Commands

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

The following parameters are provided:

Command | Command length | Reply | Length of reply area

Name	Format	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 with the indicated length.

If the user has set a non-zero reply length, any reply segments from IMS 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 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 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:

Trancode

Name	Format	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:

Trancode | Message | Message length | MOD name

Name	Format	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:

Trancode

Name	Format	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.

CMDIRNMZ - Switch to Another Conversational Transaction w. 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:

Trancode

Name	Format	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:

Trancode | Message | Message length

Name	Format	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.

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:

Trancode | Message | Message length

Name	Format	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 where it has left it, which means that the roll slot is obtained from the swap pool 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 CLEAR 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:

PCB number | Receiving area | Area length

Name	Format	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 PCBNUM 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:

MOD name

Name	Format	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:

Message area | Message area length

Name	Format	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:

Message area | Message area length

Name	Format	Type	Comment
Message area		Input	
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 including the LLZZ bytes into the message area. 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.

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:

Offset | Length | Area

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

Sample Programs: NIPSGSPA and NIPSPSPA

CMIMSID - Get MVS Subsystem ID

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

The following parameter is provided:

IMSID

Name	Format	Type	Comment
IMSID	(A4)	Output	

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

The module CMIMSID depends upon an internal IMS control block. Therefore, it is an IMS 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:

IMSID | SUFFIX | APPLGNAM | APPLNAM | NRENT | NNONR

Name	Format	Type	Comment
IMSID	(A4)	Output	The IMS 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 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:

PSB name | PCB name | PCB address

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

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:

Destination

Name	Format	Type	Comment
Destination	(A8)	Input	

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

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. No check is made regarding the IO-PCB status code after the insert call is issued to IMS/TM. In this way, you can send MFS-formatted output messages back to the originator of the input message.

The following parameters are provided:

Message area | Message length | MOD name

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

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:

Offset | Length | Data

Name	Format	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:

Transaction code | Offset | Length | Uoffset | PSB name | Number of PCBs

Name	Format	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:

Destination | Message | Message length

Name	Format	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 parameter of the NIMTRNTG macro 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:

Destination | Message | Message length

Name	Format	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 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 parameter of the NIMTRNTG macro 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:

Printer name

Name	Format	Type	Comment
Printer name		Input	

Before calling CMSNFPRT, use the Natural 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 222 and creates a dump.

Parameters: None

Sample Program: None.

CMTRNSET - Insert SPA via Alternate PCB

When the Natural session is terminated normally, the Natural IMS 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:

Trancode

Name	Format	Type	Comment
Trancode		Input	

Sample Program: NIPSEOSS

NIIDDEFS - Deferred Switch to Foreign Transaction

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:

Transaction code | Message | Message length | MOD name | Transaction type

Name	Format	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 PURG 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 PURG call.

The following parameters are provided:

Destination | Message | Message length

Name	Format	Type	Comment
Destination		Input	
Message		Input	
Message length	(B4)	Input	
TP PCB number	(B4)	Input	Optional

It is your responsibility to issue the PURG 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 parameter of the NIMTRNTG macro 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:

Transaction type

Name	Format	Type	Comment
Transaction type	(A4)	Input	Possible values: "CONV" for conversational, "NONC" for non-conversational.

Natural under IMS/TM - User Exits

This document 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
- NIIXSTAR
- NIIXMSSP
- NIIXSSTA
- NIIXISRM
- NIIXISRT
- NIIXTGU0
- NIIXJESA
- NIIXPRT0
- NIIXRFNU
- NIIXTGNO

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
- **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
- **Further information** - refer to the following topics:
Environments | Components | Configuration Macros | Service Programs | Service Modules | Special Functions | Recovery Handling

NIIXACCT

The exit is called before an accounting record is written to the IMS 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 interface to terminate the Natural session. Any other non-zero value in register 15 forces the interface to issue the Natural IMS 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.

NIIXMSSP

The exit is called only if the multi-session feature is in use. It builds the first 7 bytes of the session identification. By default, Natural under IMS compresses the LTERM names to 7 characters.

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 interface to terminate the Natural session. Any other non-zero value in register 15 forces the interface to issue the Natural IMS interface error 3509 with the reason code containing the value in register 15.

NIIXSRM

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. NIIXTGN0 receives control immediately after the message segment is retrieved, regardless of the status code.

Natural under IMS/TM - Special Functions

This document describes the use of special functions available with the Natural IMS Interface. The following topics are covered:

- Prerequisites
 - Accounting
 - Monitoring
 - Broadcasting
 - Multi-Session Feature
 - Server Environment
-

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
 - **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
 - **Further information** - refer to the following topics:
Environments | Components | Configuration Macros | Service Programs | Service Modules | User Exits | Recovery Handling
-

Prerequisites

Some of these functions require the 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 parameter ACTACTV to "YES".

With each terminal I/O, information about the specific Natural session is written to the IMS log or to SMF, depending on the setting of the Natural IMS parameter ACTLOG.

- If the ACTLOG parameter 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 utility DFSERA10. The header string is defined by the environment table parameter ACTAHDR.
- If the ACTLOG parameter is set to "LOG", the accounting record is written to the IMS/TM log using the LOG call. With the Natural IMS parameter ACTARID, you specify the log code to be used.
- If the ACTLOG parameter is set to "SMF", the accounting record is written to SMF using the Authorized Services Manager. With the Natural IMS parameter ACTARID, 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 ID of the IMS system in which the user is active,
- LTERM name of the IMS 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. 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 the Natural IMS parameter CMBSIZE 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 parameter MONACTV to "YES" 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 the SYSTP 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 parameter BROACTV to "YES" 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 system as the sender of the message;
- all users of the Natural subsystem within the same IMS 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 system as the sender of the message, but additionally restricted to a Natural application;
- all users of the Natural subsystem within the same IMS 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.



When a broadcast message is deleted or created, all expired messages are deleted as well.

Multi-Session Feature

Functionality of the Multi-Session Feature

The multi-session feature is only available in dialog-oriented environments. It is activated by setting the Natural IMS parameter MSACTV to "YES" and allows to run up to nine simultaneous Natural sessions on the same logical terminal.

Creating and resuming Natural sessions is controlled using PF keys. The Natural IMS parameters MSCRKEY and MSRSKEY define the create and resume keys.

- If the create key is pressed, the current Natural session is suspended and a new Natural session is created.
- If the resume key is pressed, any input field may contain the string x , with x between 1 and MSSESMX. In this case, the currently active session is suspended and the specified session is resumed. If x is greater than MSSESMX, the input is passed to the active application.
- If the resume key is entered without any input, the next suspended session is resumed in a wrap-around manner. If, for example, the active session is session number 4, the next suspended session in the range 5 through MSSESMX or 1 through 3 is resumed. If there is no suspended session, the input is passed to the active application.
- If the session to be resumed is using a different transaction code than the current session, an implicit program-to-program switch is done in order to resume the suspended session with the correct transaction code.
- If conversational and non-conversational environments are used in parallel, the modnames NIIMODNC and NIIMODMS are used for switching between the environments. The format definitions are delivered in the source modules NIIMODMS and NIIMODNC and have to be defined to MFS using MFSUTL.
- If the active session is terminated, the next suspended session (if there is any) is resumed in the same order as with the resume key.
- If the active session is terminated by a program-to-program switch, all suspended sessions are discarded.
- If the Natural IMS parameter MSMAX is set to "2" (two parallel sessions are allowed), the create key must be identical with the resume key. In this case, pressing the create key creates a second session if only one session is active, if not, it switches the session.

Note:

The Natural IMS Interface Version 2.2 mode is still supported for compatibility reasons, but is no longer documented. The Version 2.2 mode is activated by setting the Natural IMS parameter MSCMPTB to "YES".

Session ID

The session ID is the internally used unique identification of a session. It has the form `XXXXXXxY`, where `XXXXXXx` is the prefix and `Y` is the session number in the form of a numeric digit in the range of 1 to 9.

The session prefix is built by compressing the logical terminal name into a 6-byte binary number `XXXXXX` and by setting `x` to binary zero.

If the compression algorithm is not suitable for the LTERM names used (error 3635 is issued in this case), the user exit NIIXMSSP must be used in order to build a unique 7-byte prefix of the session ID.

Multi-Session Database

The multi-session database is a HDAM root-only database which contains the Natural-reserved area of the SPA for each suspended session.

The model DBD for the description of the multi-session database is delivered in the source module NIIMSDBD and must be defined to IMS/TM.

The DBD name must be specified in the Natural IMS parameter module using the parameter MSDBD. The PCB number must be specified in the transaction code table using the parameter MSPCB of macro NIMTRNTG.

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 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 Interface* (in the *Natural Installation Guide for Mainframes*).

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 dataset.

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 interface, refer to *Sender Destination* in the section *Natural under IMS/TM - Environments*.



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=YES is specified on the NIMDRIV macro. For details about the usage of the transaction code, refer to the NIMBOOT macro in the section Natural under IMS/TM - Environments).

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 "NIPSRVR" 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=S0201
S0201 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=S0201
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.

Natural under IMS/TM - Recovery Handling

This section describes recovery handling in the Natural IMS Interface. The following topics are covered:

- System and User Abends
 - Non-Recoverable Errors
 - Recoverable Errors
-

Additional Information

- **Natural IMS/TM Error Codes** - for a list of the error codes and messages that may be issued by the Natural IMS Interface (NII) refer to Natural IMS/TM Error Codes in the Natural Messages and Codes documentation.
 - **Installation** - refer to Installing the Natural IMS Interface in the Natural Installation Guide for Mainframes.
 - **Further information** - refer to the following topics:
Environments | Components | Configuration Macros | Service Programs | Service Modules | Special Functions | User Exits | Special Functions
-

System and User Abends

The Natural IMS 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 LE370=YES is specified on the NIMDRIV macro 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 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 interface issues a NII error and terminates the session. The error message is also written to the IMS 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 log, set the ERRLHDR parameter of the NIMPARM macro explicitly to null, i.e. 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 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.

Natural under TIAM

This document describes how to run Natural under TIAM.

The following topics are covered:

- Structure of the Natural TIAM Interface
- Parameters in Macro NAMTIAM
- Common Memory Pools under TIAM
- Natural Shared Nucleus

For information on the following topics, refer to:

- Installing the Natural TIAM Interface (in the Natural Installation Guide for Mainframes)
 - Natural under BS2000/OSD (in the Natural Operations documentation for Mainframes).
-

Structure of the Natural TIAM Interface

The Natural TIAM interface consists of two components:

- the non-reentrant front-end part
- the reentrant part "NATRENT" (default)

Both components are elements of the macro NAMTIAM and are generated with two separate assembly runs; see also Parameters in Macro NAMTIAM, parameter CODE.

The **front-end part** is generally linked with the Adabas interface module ADALNK to form the initialization routine which is run once only during the establishment of a Natural under TIAM session. During the initialization phase, based on the operand values of the corresponding parameters, various functions, for example, the establishment/connection to the Natural buffer pool, loading or linking of the Natural nucleus, establishing the physical terminal buffer, are executed. The front-end part must be loaded for each user (task).

The **reentrant part** NATRENT is linked as a modular element to the Natural nucleus and contains various entry points for TP system dependent routines (memory management, terminal communication, etc.). If a shared Natural nucleus is to be used, the generated NATRENT module must be linked to the front-end part.

The **Natural nucleus** is completely environment-independent (shared code) and must be loaded only once for all users.

Parameters in Macro NAMTIAM

The macro NAMTIAM has to be generated twice: for the front-end part of the Natural TIAM Interface and for the reentrant part. For which part it is generated is determined by the parameter CODE in the NAMTIAM macro.

For the generation of the front-end part and the reentrant part, a label can be provided in the macro call to NAMTIAM. This label defines the CSECT name, under which the module is stored in the module library. If no label is specified, the name for the front-end part is "NATFRONT" and the name for the reentrant part is "NATRENT".

Example of NAMTIAM Macro for Front-End Part:

```
NATTESTF NAMTIAM CODE=FRONT,
  NUCNAME=NB2RENT,
  PARMOD=31
```

In this example, the CSECT name of the front-end part is defined as NATTESTF.

Example of NAMTIAM Macro for Reentrant Part:

```
NATTESTR NAMTIAM CODE=RENT,
  CLRKEY=K4,
  PARMOD=31
```

In this example, the CSECT name of the reentrant part is defined as NATTESTR.

Parameters:

The individual parameters which can be specified in macro NAMTIAM are explained below:

ADACOM | ADDBUFF | APPLNAM | ATTKEY | CLRKEY | CODE | CURPRO | DELETE | DYNPAR |
 HCASK | ILCS | LF | LINK | LINK2/LINK3/LINK4 | NUCNAME | PARMOD | PFK | REFRKEY | REQMLC
 | SYSDTA | TRACE | TTYLS | TTYPS | T975X | UMODE | USERID |

ADACOM

Possible values:	ADAUSER/ADALNN/ADABAS/ADALNK
Default value:	ADALNK

This parameter applies to the generation of the front-end part. It determines which Adabas link module is to be used.

ADACOM=ADAUSER	The module ADAUSER is linked to the front-end part (Adabas versions lower than 7.1).
ADACOM=ADALNN	The module ADALNN is linked to the front-end part if Multi-Pass is in use (Adabas versions lower than 7.1).
ADACOM=ADABAS	The modules ADAUSER + SSFB2C are linked to the front-end part (Adabas Version 7.1 and higher).
ADACOM=ADALNK	The module ADALNK is linked to the front-end part (Adabas versions lower than 7.1) or the modules ADALNK, ADAL2P + SSFB2C are linked to the front-end part (Adabas Version 7.1 and higher).

ADDBUFF

Possible values:	1 to 8/no operand
Default value:	No operand

This parameter applies to the generation of the front-end part.

It determines the additional number of pages (4-KB units) for the terminal I/O buffer.

APPLNAM

Possible values:	<i>name</i>
Default value:	NATTIAM

This parameter applies to the generation of the front-end part.

name is the name (maximum 8 characters) of the Natural TIAM application. This name is part of the serialization ID when the Natural TIAM task is initialized.

ATTKEY

Possible values:	ON/OFF
Default value:	OFF

This parameter applies to the generation of the reentrant part.

ATTKEY=ON	Pressing the K2 key on your terminal is intercepted by an STXIT routine. Natural creates an ATTENTION INTERRUPT and returns a NAT1016 error message.
ATTKEY=OFF	Pressing the K2 key leads to a normal BS2000/OSD interrupt.

CLRKEY

Possible values:	K1 to K4, F1 to F4, DUE2
Default value:	K4

This parameter applies to the generation of the reentrant part.

This parameter can be used to define an alternate CLEARKEY in addition to LSP and DUE1.

CODE

Possible values:	FRONT/RENT
Default value:	FRONT

This parameter applies to the generation of both the front-end and reentrant parts.

It determines which part of the Natural TIAM Interface is to be generated.

CODE=FRONT	Indicates the generation/assembly of the front-end part.
CODE=RENT	Indicates the generation/assembly of the reentrant part.

CURPRO

Possible values:	ON/OFF
Default value:	ON

This parameter applies to the generation of the front-end part.

It controls whether the cursor can be positioned to a protected field.

CURPRO=ON	The cursor <i>cannot</i> be positioned to a protected field.
CURPRO=OFF	The cursor can also be placed in a protected field (for example, for field-specific help functions).

DELETE

Possible values:	ON/OFF
Default value:	ON

This parameter applies to the generation of the reentrant part.

DELETE=ON	The setting of the profile parameter DELETE (see DELETE - Deletion of Dynamically Loaded Programs in the Natural Parameter Reference documentation in the Natural parameter module determines whether dynamically loaded non-Natural programs are unloaded at the end of the Natural program in which they are loaded or whether they are unloaded when command mode is entered.
DELETE=OFF	A dynamically loaded non-Natural program once loaded is kept for the duration of the whole Natural session.

DYNPAR

Possible values:	NO/YES/DIALOG/SYSDTA/SYSIPT/FILE
Default value:	NO

This parameter applies to the generation of the front-end part.

DYNPAR=DIALOG	The dynamic parameters are read from terminal input.
DYNPAR=YES	The dynamic parameters are read from terminal input. YES has the same effect as DIALOG; it is kept for compatibility reasons.
DYNPAR=SYSDTA	The dynamic parameters are read from SYSDTA.
DYNPAR=SYSIPT	The dynamic parameters are read from SYSIPT.
DYNPAR=FILE	The dynamic parameters are read from a sequential file. The input of this SAM file is interpreted as one single text string, which means that the individual entries must be separated from each other by a comma, even at the end of a line. Such a parameter file must be defined with a FILE command by using the LINK parameter CMPRMIN. See also the example given below.
DYNPAR=NO	No dynamic parameters are read from terminal input.

Example for DYNPAR=FILE:

```
/FILE NAT.PARAMS, LINK=CMRMIN
```

HCASK

Possible values:	ON/OFF
Default value:	ON

This parameter applies to the generation of the reentrant part.

It determines whether a user is asked to specify an output device each time he or she produces a hardcopy (with terminal command %H).

HCASK=ON	The user is asked to specify a device for each hardcopy.
HCASK=OFF	The device used for the previous hardcopy is used again.

ILCS

Possible values:	CRTE/YES/NO
Default value:	NO

This parameter applies to the generation of the reentrant part.

ILCS=CRTE	3GL subprograms are invoked with common runtime environment convention. To be able to do so, the ILCS initialization routine IT0SL# must be linked to the Natural front-end, as shown below: <pre>INCLUDE IT0SL#,SYSLNK.CRTE.010 RESOLVE,SYSLNK.CRTE.010</pre>
ILCS=YES	3GL subprograms are invoked with enhanced ILCS linkage convention. To be able to do so, the ILCS initialization routine IT0INITS must be linked to the Natural front-end, as shown below: <pre>INCLUDE IT0INITS,SYSLNK.ILCS RESOLVE,SYSLNK.ILCS</pre>
ILCS=NO	Standard processing applies.

LF

Possible values:	X'zz'
Default value:	X'25'

This parameter applies to the generation of the front-end part.

With this parameter you specify the control character to be used for line advance when printing on the local printer.

LINK

Possible values:	<i>name</i> (<i>name,name,...</i>)
Default value:	none

This parameter applies to the generation of the front-end part.

The *name(s)* of programs and modules that are called from Natural programs and linked with the non-reentrant part must be specified with this parameter. Conversely, the programs and modules whose names are specified must be linked with the non-reentrant part, otherwise the application is put into status SYSTEMERROR and all users are rejected with an error message.

A "TABLE" macro call is performed for the specified programs and modules, which enters their load addresses into the dynamic loader's link table. It is therefore not necessary to dynamically load these programs when they are called by Natural programs.

Example:

```
LINK=PROG1
LINK=(PROG1,PROG2,MODUL111)
```

LINK2/LINK3/LINK4

Possible values:	<i>name</i> (<i>name,name,...</i>)
Default value:	none

These parameters apply for the generation of the front-end part.

The parameters LINK2, LINK3 and LINK4 are an extension of the LINK parameter. Since an operand definition cannot be longer than 127 characters (including parentheses), these parameters are provided for cases where the operand of parameter LINK would be too long. The syntax is analogous to that of LINK.

<p>Examples:</p> <pre>NAMTIAM LINK=(PROG1,PROG2,...), LINK2=(PROG54,...) NAMTIAM LINK=(PROG1,PROG2,PROG3,PROG4)</pre>
--

NUCNAME

Possible values:	<i>name</i>
Default value:	NB2RENT

This parameter applies to the generation of the front-end part.

With this parameter you specify the name of the bounded, reentrant Natural module. You must use this name for the Natural pool and load information in macro ADDON (BS2STUB assembles macro ADDON).

PARMOD

Possible values:	<i>(nn,loc)</i>
<i>nn:</i>	24/31
<i>loc:</i>	BELOW/ABOVE
Default value:	(31,ABOVE)

This parameter applies to the generation of both the front-end and reentrant parts.

- The first part of this parameter (*nn*) is used to define an addressing mode (24-bit or 31-bit mode) for the Natural TIAM application.
- The second part of this parameter (*loc*) is used to define the front part location of the Natural TIAM application. If you load the front part of the application above 16 MB, this must be defined in the front part's link procedure as follows:
LOADPT=*XS or
LOADPT=X'address'

Example:

```
/EXEC TSOLINK

  PROG NATvrs, FILENAM=NATvrs, LOADPT=*XS, ...

  TRAITS RMODE=ANY, AMODE=31

  INCLUDE....

/* PARMOD=(nn,loc) MUST BE IDENTICAL IN THE FRONT-END AND REENTRANT
PARTS
```

PFK

Possible values:	(KN,y) (KO,y) (KS,y) OFF
Default value:	(KS,L)

This parameter applies to the generation of the front-end part.
It is used to set one of the following function-key modes:

KN	Either literals "%K1" to "%K20" and send-key code "DÜ1" or send-key codes "F1" to "F20" are loaded to the function keys; this depends on the device type.
KO	The literals "01" to "20" and send-key code "F5" are loaded to the function keys.
KS	The literals "A" to "T" and send-key code "F5" are loaded to the function keys; in addition, with every output message a dummy field is generated at the last two positions of the screen. This dummy field is used to receive and pass the key value.
OFF	No function key mode is generated.
y	This can be either "L" or "N": - "L" means that the function keys are loaded; - "N" means that the corresponding mode is activated, but function keys will not be loaded.

REFRKEY

Possible values:	K1 to K14 / NO
Default value:	K14 (keys ESC + ":")

This parameter applies to the generation of the reentrant part.

It can be used to define a function key. If this function key is pressed, the last full Natural screen is refreshed. Thus it is possible to continue the dialogue with Natural after the screen has been overwritten by messages from the operator or the operating system.

The send-key code is not passed to the Natural application. The interface sets the Natural key code to "ENTER". The key defined with the REFRKEY parameter must be different from the one defined with the CLRKEY parameter.

REQMLOC

Possible values:	RES/BELOW/ABOVE
Default value:	RES

This parameter applies to the generation of both the front-end and reentrant parts in 31-bit mode (PARMOD=31).

It determines where the requested Natural work areas are to be allocated via request memory by the system.

REQMLOC= BELOW	All areas are requested below 16 MB.
REQMLOC= ABOVE	All areas are requested above 16 MB.
REQMLOC=RES	All areas are requested depending on the location of the reentrant part.

The REQMLOC parameter corresponds to the LOC parameter of the BS2000/OSD system macro REQM.

SYSDTA

Possible values:	PRIMARY/SYSCMD
Default value:	PRIMARY

This parameter applies to the generation of the front-end part.

SYSDTA=PRIMARY	After reading of dynamic Natural parameters from SYSDTA, SYSDTA is set to SYSDTA=(PRIMARY).
SYSDTA=SYSCMD	After reading of dynamic Natural parameters from SYSDTA, SYSDTA is set to SYSDTA=(SYSCMD).

TRACE

Possible values:	(<i>nn,ll</i>)
<i>nn:</i>	01...99
<i>ll:</i>	71...132
Default value:	(99,71)

This parameter applies to the generation of the reentrant part.

With this parameter, you specify the number of a trace file and the maximal length of a trace print record. *nn* is the number for the SYSLST*nn* trace file and *ll* is the maximal length in characters of a trace print record.

If any external Natural trace function is active, the trace records will be written to SYSLST*nn*. In this case, the Natural TIAM driver creates the following trace file:

Example:

```
NATURAL.TRACE.TIAM.TTTT,SPACE=(30,3)
  SYSFILE SYSLSTnn=NATURAL.TRACE.TIAM.TTTT

/* TTTT is the task sequence number
```

Before the Natural TIAM session is terminated, the trace file will be closed as follows:

```
SYSFILE SYSLSTnn=(PRIMARY)
```

TTYLS

Possible values:	<i>nn</i>
Default value:	80

This parameter applies to the generation of the front-end part.

With this parameter you can adjust Natural's physical line length to different paper formats used with a telex machine.

nn specifies the physical line size for TTY devices.

TTYPS

Possible values:	<i>nn</i>
Default value:	24

This parameter applies to the generation of the front-end part.

With this parameter you can adjust Natural's physical page size to different paper formats used with a telex machine.

nn specifies the physical page size (number of lines) for TTY devices.

T975X

Possible values:	9750/9755/9756/9763
Default value:	9750

This parameter applies to the generation of the front-end part.

It is used to determine for which device types messages are to be optimized when using data stations which were generated in PDN as 9750.

UMODE

Possible values:	(S, <i>y</i>)
Default value:	(S,E)

This parameter applies to the generation of the front-end part.

Operands:

S	Mode of operation = single; that is, one Natural session can be started from a terminal.
y	Language indicator for the logoff message. D=Danish E=English F=French G=German I=Italian N=Dutch S=Spanish

USERID

Possible values:	YES/SYSTEM/NO/USER
Default value:	USER

This parameter applies to the generation of the front-end part.

USERID=SYSTEM or USERID=YES	The Natural user ID is created by using the BS2000/OSD user ID.
USERID=USER or USERID=NO	The Natural user ID is created by using the job name; that is, the "/.JOBNAME" of the LOGON command. If no BS2000/OSD job name has been specified with the LOGON command, the Natural user ID is created as with USERID=SYSTEM or YES.

Common Memory Pools under TIAM

You use macro ADDON (which assembles module BS2STUB) either to generate the local common memory pools, or to define attachment to the global common memory pools.

The programs CMPSTART and CMPEND start and stop *global* common memory pools. They are described in the section Global Common Memory Pools (in the Natural Operations documentation for Mainframes).

A Natural TIAM application needs the following common memory pools:

- **Natural load pool**
The linked reentrant part of Natural is loaded into this common memory pool.
- **Natural buffer pool**
The executable Natural programs and the Natural global data areas are loaded into this common memory pool. Those compiled Natural programs whose objects are reentrant are executed from this memory pool.
- **Natural/Adabas nucleus communication memory pool**
Natural connects to an additional common memory pool which is established by Adabas during startup.

The sum of the memory assigned to common memory pools, as well as the front-end work area, must completely fit into the virtual user address space.

If the Adabas pool exceeds the user address space, error message 148 is produced during the OP command execution. At the beginning of the session, Natural issues the error message NAT8148 and in the following session termination with the message NAT9989 (incorrect system file).

Natural Shared Nucleus

For TIAM applications, it is possible to use a common shared Natural nucleus. The rules that apply in this case are documented in the section Natural Shared Nucleus under BS2000/OSD (in the Natural Operations documentation for Mainframes)

Natural under TSO

This document contains special considerations that refer to Natural under TSO in the operating system OS/390.

The following topics are covered:

- General Information about the Natural TSO Interface
- Natural TSO Interface Generation Parameters
- Datasets Used by Natural under TSO
- Issuing TSO Commands from Natural under TSO

See also:

- Installing the Natural TSO Interface (in the Natural Installation Guide for mainframes).

General Information about the Natural TSO Interface

The Natural TSO interface (NATTSO) consists of a number of service routines interfacing with the OS/390 operating system.

NATTSO is supplied as a source module and can be customized to meet your requirements; see also Installing the Natural TSO Interface (described in the Natural Installation Guide for Mainframes). You can either assemble and link NATTSO to the Natural nucleus or you can run it separately by connecting it with a shared nucleus.

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 below).

Natural TSO Interface Generation Parameters

The NATTSO macro contains several generation parameters to change Natural TSO interface's internal defaults.

These parameters are:

ABEXIT | ALTSCRN | FACOM | LBPNAME | LE370 | NDBFSRV | SUBPOOL | SWAPKEY | TTYTYPE

ABEXIT - Abend Processing Mode

This parameter specifies the mode of abend processing within Natural.

ABEXIT=ESTAE	Natural intercepts all abends and issues the appropriate error messages. This is the default value.
ABEXIT=SPIE	Only program checks (S0Cx abends) are intercepted as they used to be with Natural Version 2.1.
ABEXIT=NONE	Natural does not intercept any abends or program checks at all. This value corresponds to the Natural profile parameter DU=FORCE.

ALTSCRN - Session Screen Mode

This parameter specifies whether the 3270 alternate screen size is to be used. There are 2 sets of screen heights/widths from the VTAM LOGMODE definition for the terminal, default and alternate screen size. Usually, the default screen size is 24 lines and 80 columns. The alternate screen size depends on the 3270

terminal model (2, 3, 4 or 5).

ALTSCRN=YES	The alternate screen size is to be used. This is the default value.
ALTSCRN=NO	The default screen size is to be used.

With Natural profile parameter TMODEL, the screen size can be overwritten.

FACOM - Use of FACOM Operating System

This parameter specifies whether the FACOM operating system is to be used.

FACOM=NO	FACOM is not used. This is the default value.
FACOM=YES	FACOM is to be used. Specific code is generated to support FACOM.

LBPNAME - Sharing of Local Buffer Pools

This parameter controls the sharing of the local buffer pools. It defines the *name* of the shared buffer pool environment and is used to locate and synchronize the local buffer pools.

LBPNAME= <i>name</i>	<i>name</i> can be 1-8 characters long.
LBPNAME=	The default value is none, that is, local buffer pools are not shared.

When running multiple Natural sessions under OS/390 in a batch or TSO environment concurrently, for example, when running a Natural RPC server, each session allocates storage for separate local buffer pools. Except for the Natural OS/390 batch server, the local buffer pools are not shared per default, that is, if the different sessions use the same Natural objects, these have to be loaded for each session separately. If *name* is specified, all local buffer pools will be shared by the different Natural sessions.

LE370 - Use of IBM Language Environment

This parameter specifies whether Natural is to run in the IBM Language Environment (LE).

LE370=YES	You can call external subprograms according to the LE calling conventions.
LE370=NO	You can only call main programs of the language environment. This is the default value. This means, that a new LE enclave is created and terminated for each CALL statement.
LE370=POSIX	You can call external subprograms according to the LE calling conventions with POSIX semantics, i.e. the LE is initialized with runtime option POSIX(ON).

For more information about Natural running with the IBM Language Environment, refer to Miscellaneous > LE Subprograms (in the Natural Operations Manual).

NDBFSRV - Natural for DB2 File Server

This parameter specifies whether the Natural for DB2 file server is to be used.

NDBFSRV=NO	The Natural for DB2 file server will not be used. This is the default value.
NDBFSRV=YES	The Natural for DB2 file server is invoked at each terminal I/O.

SUBPOOL - Storage Subpool for GETMAIN Requests

This parameter defines the storage subpool for GETMAIN requests.

SUBPOOL= <i>nnn</i>	Possible value for <i>nn</i> : 0 - 127. The default value is 0.
---------------------	--

SWAPKEY - TSO/ISPF Split Screen Feature Support

This parameter defines the TSO/ISPF split screen swap key for Natural, which is assigned to PF9 on most of the panels of Software AG product Natural ISPF (Integrated Structured Programming Facility).

SWAPKEY=PF <i>nn</i> SWAPKEY=PA <i>nn</i>	Defines the PF- or PA-key which shall be used to swap to the next TSO/ISPF session. Valid keys are PF1 to PF24 and PA1 to PA3. Note: The specified key cannot be used by Natural applications. Usually, the SWAP command in TSO/ISPF is assigned to the PF9 key on most TSO/ISPF panels. Split screen support requires the TSO/ISPF interface module ISPLINK from the ISPF load library to be included in the link step for NATTSO.
SWAPKEY=	No swap key is defined, i.e. no split screen feature support is generated. This is the default value.

TTYTYPE - Default Device Type for TTY Terminals

This parameter defines the default device type for teletype terminals (TTY or TWX).

TTYTYPE= <i>name</i>	<i>name</i> defines the default device type for teletype terminals. The specified type must be defined in the NATCONFIG (Natural configuration) module. The default value is TTY.
----------------------	---

TTYTYPE can be overwritten by the TTYPE profile parameter or by the "%T=" terminal command during the session.

Datasets Used by Natural under TSO

The following datasets are required if certain functions are used during a Natural TSO session:

CMEDIT	Software AG Editor Work File
CMHCOPY	Hardcopy Print Output
CMPLG	Dynamic Profile Parameter Report Output
CMPRMIN	Dynamic Profile Parameter Input
CMPT <i>nn</i>	Additional Reports 01-31
CMTRACE	External Trace Output
NATRJE	Job Submit Output
STEPLIB	Load Library for External Modules
CMWKF <i>nn</i>	Work Files 01-32

These datasets are described below.

Unless otherwise stated below, the default DCB RECFM/LRECL information is as follows:

RECFM=FB and LRECL=80 for sequential input datasets

RECFM=FBA and LRECL=133 for sequential output datasets

CMEDIT - Software AG Editor Work File

The Software AG editor work file VSAM dataset 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 (in the Natural Installation for Mainframes documentation).

CMHCOPY - Hardcopy Print Output

The default name of the hardcopy print output dataset 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 dataset. The default dataset name CMHCOPY implies CLOSE=FIN for the hardcopy print dataset, that is, after the dataset is opened for output, any subsequent change of the hardcopy print output dataset name is not honored. If a different name is defined at open time, the hardcopy dataset will be closed upon the next terminal I/O.

During the session, the hardcopy dataset can be released and reallocated (before open or after close) by the by dynamic allocation (see Natural Application Programming Interface USR2021).

CMPLOG - Dynamic Profile Parameter Report Output

If profile parameter PLOG=ON is set and dataset CMPLOG is available, the evaluated dynamic profile parameters are written to this dataset during session initialization. If dataset 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 dataset 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.

CMPR Tnn - Additional Reports 01-31

These datasets 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 PRINT profile parameter or by the NTPRINT macro in the Natural parameter module. The print file dataset names can be overwritten by subparameter DEST.

CMTRACE - External Trace Output

If profile parameter ETRACE=ON is set or the equivalent terminal command %TRE+ was issued, any Natural trace output during the session is written to the CMTRACE dataset. To define the Natural components that shall be traced, the profile parameter TRACE is required.

If dataset 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 dataset 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.

CMWKF nn - Work Files 01-32

These datasets can be used by Natural work file statements like READ WORK 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 dataset, the defaults are defined by the WORK profile parameter or by the NETWORK macro in the Natural parameter module. The work file dataset 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.

Natural under UTM - Overview

The Natural UTM Interface documentation is divided into four parts and covers the following topics:

Part 1

- Structure of the Natural UTM Interface
- Formatting Messages - FREXIT
- Embedding Natural in a UTM Application
- Common Memory Pools
- Other Storage Areas
- Generating KDCROOT
- Defining the UTM Resources - KDCDEF
- UTM DC-Transaction Exit Routine NUERROR
- UTM Startup Function
- UTM Shutdown Function

Part 2

- NATUTM Macro Keyword Parameters
- NATUTM Macro Entries
- NURENT Macro Keyword Parameters

Part 3

- User Exits
- Asynchronous Transaction Processing under UTM
- Printing under UTM
- Calling Non-Natural Programs
- Calling UTM Chained Partial Programs
- Calling Adabas from Non-Natural Programs in a Natural UTM Application
- Terminating a UTM Task Abnormally

Part 4

- Accounting for Natural UTM Applications
- Utility Programs for Use with Natural under UTM
- Software Exchange
- UTM TACCLASS Concept (Priority Control)
- Generating a Natural UTM Application
- Optimizing Natural UTM Applications
- Several Applications with one Common Natural
- Entering and Defining Dynamic Natural Parameters
- UTM User Restart
- Adabas Priority Control

See also:

- Installing the Natural UTM Interface (in the Natural Installation Guide for Mainframes)
- Natural under UTM Error Messages (in the Messages and Codes documentation for Mainframes).
- Error Messages from the Natural Swap Pool Manager Valid under CICS and UTM (in the Messages and Codes documentation for Mainframes).
- Natural under BS2000/OSD (in the Natural Operations documentation for Mainframes).

Notation *vrs* or *vr*

If used in the following document, the notation *vrs* or *vr* stands for the relevant *version, release, system maintenance level* numbers.

Natural under UTM - Part 1

This part of the Natural UTM Interface documentation covers the following topics:

- Structure of the Natural UTM Interface
- Formatting Messages - FRETIT
- Embedding Natural in a UTM Application
- Common Memory Pools
- Other Storage Areas
- Generating KDCROOT
- Defining the UTM Resources - KDCDEF
- UTM DC-Transaction Exit Routine NUERROR
- UTM Startup Function
- UTM Shutdown Function

Installation - refer to Installing the Natural UTM Interface in the Natural Installation Guide for Mainframes.

Notation *vrs* or *vr*>**Notation *vrs* or *vr***: If used in the following document, the notation *vrs* or *vr* stands for the relevant *version*, *release*, *system maintenance level* numbers.

Structure of the Natural UTM Interface

The Natural UTM Interface consists of the macros NATUTM, BS2STUB and NURENT and of several utility programs, which enable special requirements to be accommodated.

The macro NATUTM is used to generate the front-end part of the Natural UTM Interface to suit the particular application based on appropriate operand definitions for the keyword parameters. The default values of the parameters are chosen so that, in general, they can be used without alteration for an initial generation. The front-end part is present once per UTM task and consists principally of the following components:

- KDCROOT of UTM,
- assembled macro NATUTM,
- assembled macro BS2STUB,
- format exit module FREXIT,
- Adabas interface module.

The reentrant part of the Natural UTM Interface is generated by assembling the macro NURENT. This is linked with the reentrant part of the Natural UTM application. If a shared Natural nucleus is to be used, the generated "NURENT" module must be linked to the front-end part.

The reentrant part of the Natural UTM application consists of the following components:

- NATINV (address module) (must be included as the first module),
- Natural nucleus,
- Natural buffer pool manager,
- NURENT (CSECT name of the assembled macro NURENT),
- NATSWPMG (Natural swap pool manager),
- Natural parameter module,
- NATLAST (end definition) (must be included as the last module).

The reentrant part of the Natural UTM Interface is only present once in a Natural UTM application (reentrant) if it is loaded into class 4 storage or into a *common memory pool* in class 6 storage. The latter is recommended. A further possibility is to link the reentrant part with the non-reentrant front-end part of the Natural UTM application. The Natural and UTM macro libraries are required when assembling NATUTM, NURENT and all utility programs.

Formatting Messages - FREXIT

Natural uses its own formatting routines when sending messages to the VDU (UTM format type "-"). Messages are processed by the format exit module FREXIT (transfer from logical to physical I/O domain and vice versa, producing RESTART and LOGOFF messages, etc.). The module FREXIT must be linked with the front-end part of the Natural UTM application and it must be defined as the format exit module when generating KDCROOT or KDCDEF.

Example:

```
PROGRAM FREXIT,COMP=ASSEMB
EXIT PROGRAM=FREXIT,USAGE=FORMAT
```

The program FREXIT supports the format name "-END" for the LOGOFF message. See the description of the keyword parameter LOFFMAP of the macro NATUTM. No more UTM administration commands (KDCINF, KDCSHUT N, etc.) can be entered after the format name "-END" has been used and the LOGOFF message has been output. The LOGOFF message is output in formatted mode; however, UTM expects administration commands in line mode and therefore any input results in a syntax error. After this error message has been received, all valid administration commands can be input with the administration ID. The messages for asynchronous messages, RESTART and LOGOFF can be changed to suit specific requirements by changing the appropriate text constants in the program FREXIT.

The program FREXIT has a user exit INPTEX that can be satisfied by the utility program INPTEX. See the descriptions of the programs NATDUE and INPTEX in the section Utility Programs.

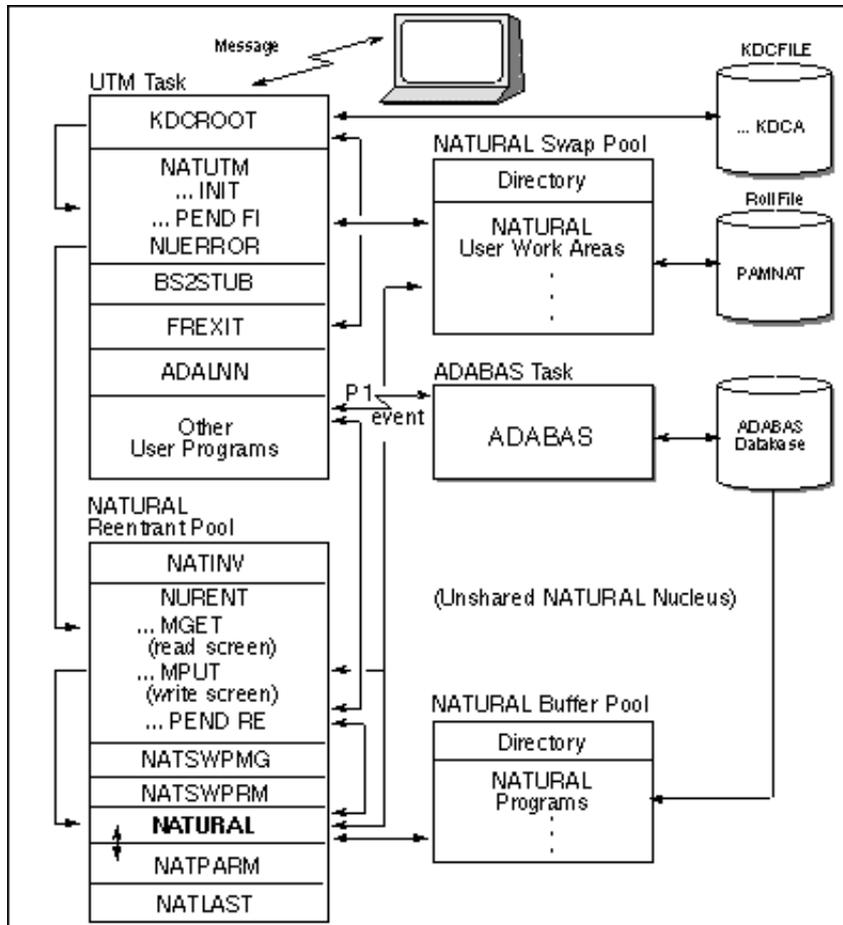
Another user exit in program FREXIT is TRMIOEX, which can be used for input/output message control.

The macro FREXIT contains the following keyword parameters:

AMSG=ASAP	If there are any "free-running" (asynchronous) messages, a further dialogue with Natural is only possible if these messages have previously been read with the command KDCOUT.
AMSG=WAIT	A further dialogue with Natural is possible even if any "free-running" messages have not yet been read with the command KDCOUT. This is the default.
KDCDISP=YES	(Default) KDCDISP is supported by a restart message with an automatic ENTER. The last screen output will be refreshed.
KDCDISP=NO	KDCDISP is supported by a restart message with a following refresh screen.

If you want to change a default operand of macro FREXIT, you must reassemble FREXIT.

Embedding Natural in a UTM Application



Common Memory Pools

The following topics are covered:

- Natural Buffer Pool under UTM
- Natural Swap Pool under UTM
- Loading Natural in a Common Memory Pool (Natural Load Pool)
- Natural Monitor Pool

Natural Buffer Pool under UTM

Natural requires a common area into which Natural programs can be read from the Adabas database and where they are also executed. This common memory pool is the Natural *buffer pool*.

You use the keyword parameters of macro "ADDON" (which assembles module "BS2STUB") either to define a local Natural buffer pool, or to define the connection to a global Natural buffer pool. For more information, see ADDON Macro in the Natural Operations for Mainframes documentation.

You use the keyword parameters of module "CMPSTART" to define a global Natural buffer pool. For more information on this module, see CMPSTART Program in the Natural Operations for Mainframes documentation.

To display statistical information about the buffer pool, use the Natural utility SYSBPM (which is described in the Natural Utilities documentation).

Natural Swap Pool under UTM

A Natural *user work area* is required for each online Natural user. This user work area must be in the computer's main store whenever the user initiates any form of dialogue transaction. To reduce the frequency with which the user work area is rolled out to the swap file and rolled in again, it is possible to set up a Natural Swap Pool. For details on the swap pool, please refer to Natural Swap Pool in the Natural Operations for Mainframes documentation.

You use the keyword parameters of macro "ADDON" (which assembles module "BS2STUB") either to define a local Natural swap pool, or to define the connection to a global Natural swap pool. For more information, see ADDON Macro in the Natural Operations for Mainframes documentation.

You use the keyword parameters of module "CMPSTART" to define a global Natural swap pool. For more information, see CMPSTART Program in the Natural Operations for Mainframes documentation.

Loading Natural in a Common Memory Pool - Natural Load Pool

The reentrant part of the Natural UTM application can be loaded in class 4 storage or linked with the front-end part of the Natural UTM application. Alternatively, it can be loaded in a common memory pool in class 6 storage. This last method is recommended. The amount of storage required in the common memory pool depends upon the size of the linked reentrant part of the Natural UTM application; this can be read from the linker listing. The following keyword parameter of macro NATUTM is used if Natural is to be loaded into a common memory pool in class 6 storage:

Parameter	Explanation
NUCNAME	This parameter specifies the name of the linked, reentrant Natural nucleus. This is also the name of the Natural load pool.

See also Keyword Parameters of Macro NATUTM.

You use the keyword parameters of macro "ADDON" (which assembles module "BS2STUB") either to define a local Natural load pool, or to define the connection to a global Natural load pool (shared Natural nucleus). For more information, see ADDON Macro in the Natural Operations for Mainframes documentation.

You use the keyword parameters of module "CMPSTART" to define a global Natural load pool (shared Natural nucleus). For more information, see CMPSTART Program in the Natural Operations for Mainframes documentation.

Natural Monitor Pool

The Natural Monitor utility requires a common memory pool for data storage. This common memory pool is allocated when the Monitor utility is activated, and released when the Monitor utility is deactivated.

You use the keyword parameters of macro "ADDON" (which assembles module "BS2STUB") either to define a local Natural monitor pool, or to define the connection to a global Natural monitor pool. For more information, see ADDON Macro in the Natural Operations for Mainframes documentation.

You use the keyword parameters of module "CMPSTART" to define a global Natural monitor pool. For more information, see CMPSTART Program in the Natural Operations for Mainframes documentation.

For details on the Monitor utility, see SYSTP Utility in the Natural Utilities documentation.

Other Storage Areas

Natural User Thread

For each UTM task a storage area with a size of MAXSIZE is generated. This area contains the Natural user area in decompressed form.

Natural User Work Area Asynchronous Write Buffer

The Natural user work area can be written out either asynchronously ("write without wait") or synchronously ("write with wait").

If the asynchronous option is used (this is the default option), a write buffer having the size of defined operand for parameter ROLLTSZ is generated for each UTM task. Using this technique, the compressed user work area is copied from the swap pool into the write buffer, the asynchronous write is started and processing can continue immediately. This option gives better performance, but at the cost of increased storage.

If roll-outs are to be performed synchronously, the keyword parameter ROLLACC must have the value "UPAM-SY". In this case, it is not necessary to allocate a write buffer. Processing is suspended until the user work area has successfully been written to the swap file.

Natural User Area for Asynchronous Transactions

A storage area of MAXSIZE is allocated for each asynchronous transaction in a Natural UTM application (Natural user work area for this transaction). It is released at the end of the transaction. The Natural swap pool is not used to store the user work area associated with asynchronous transactions. Every Natural program that runs asynchronously must end with TERMINATE; that is, the UTM DC transaction is ended with PEND 'FI(NISH)'. This applies to asynchronous transactions both within an application and between two Natural UTM applications; see also Asynchronous Transaction Processing .

Natural Roll File - LINK=PAMNAT

A PAM file is required for swapping the Natural user work areas. Writing to and reading from this file is done by physical chained PAM-I/O. However, this is only possible as long as the swap file does not cross an extent boundary. This can be checked using SPCCNTRL.

The LINK name of the Natural swap file is PAMNAT. The size of the roll file can be computed as follows:

$$NP = ([(MS + 4 + 31) / 32] * 32 * NT + 4) / 2$$

where:

NP = Size of dataset in PAM pages;

MS = Parameter ROLLTSZ in KB, rounded up to next even number;

NT = Number of terminals online.

Example:

ROLLTSZ = 80 KB (per user) Number of terminals online = 40

$$\begin{aligned}\text{Size of dataset} &= ([(80 + 4 + 31) / 32] * 32 * 40 + 4) / 2 \\ &= ([115 / 32] * 32 * 40 + 4) / 2 \\ &= ([3.59375] * 32 * 40 + 4) / 2 \\ &= (3 * 32 * 40 + 4) / 2 \\ &= 3844 / 2 \\ &= 1922 \text{ PAM pages}\end{aligned}$$

FILE statement:

```
/FILE NATUTM.SWAPFILE, LINK=PAMNAT, SPACE=(1922,96)
```

When a local swap pool is used, each Natural UTM application requires its own Natural swap file. When a user logs on to the application, the Natural UTM Interface checks whether there is sufficient space available for the new user in the Natural roll file. If there is not enough space, error message NUS0033 is output.

When a global swap pool is used, all Natural UTM applications which are connected to the same global swap pool must use the same Natural roll file.

Generating KDCROOT

The following Natural-specific definitions must be entered when generating KDCROOT for a Natural UTM application:

```

MAX  KB=400 , SPAB=8192 , NB=5120 , TRMSGLTH=5120    (see Note 1)
PROGRAM  NUSTART , COMP=ASSEMB                      (see Note 2)
PROGRAM  NUERROR , COMP=ASSEMB                      (see Note 2)
TAC  NAT , PROGRAM=NUSTART , EXIT=NUERROR          (see Note 2)
EXIT  PROGRAM=FREXIT , USAGE=FORMAT                (see Note 3)
PROGRAM  FREXIT , COMP=ASSEMB                      (see Note 3)

```

Note

- 1 The area needed for the UTM KB has a minimum length of 400 bytes. The necessary KB length for operand `KB=nnn` in the `MAX` parameter of `KDCDEF` must be calculated as follows:

Fixed KB length is 400 bytes

- + length of KB user extension (keyword parameter `KBUSEXT`)
 - + length of dynamic parameter save area (keyword parameter `SVDYPRM`)
 - + (only if `MULTI-PASS` is used) length of session key areas, which has to be calculated as follows: $n*72$ where n is the number of parallel session minus 1
- The UTM I/O areas `NB` and `TRMSLGTH` need a length of 5120 bytes.

- 2 In a Natural UTM application there is as a rule only one user-specific UTM partial program.

This program is the front-end part of the Natural UTM Interface, which must be defined in the adequate parameters of `KDCDEF` under the name specified in the operand of the keyword parameter `CSECT` of macro `NATUTM` (default = `NUSTART`).

Any number of UTM transaction codes can be assigned, providing the naming rule is observed.

The name of the DC transaction exit routine `NUERROR` must be defined for the front-end part of the Natural UTM Interface and for each other UTM partial program.

- 3 The format exit module `FREXIT` must be defined with the parameters `EXIT` and `PROGRAM`.

All other definitions relating to the generation of `KDCROOT` are either specific to UTM or else they are dependent upon the values defined in the operands of the appropriate keyword parameters of macro `NATUTM`.

Defining the UTM Resources - KDCDEF

The following Natural-specific points must be observed when defining the UTM resources:

Special Definition for Type 9755/9756 Terminals

The TERMN operand of the PTERM command must be set to the value "X1" or "FG" for 9755-type terminals and to the value "X2" for 9756-type terminals. These are special values and not described in the appropriate table in the Siemens UTM documentations.

For all other types of terminals, the TERMN operand must be set to the value shown in the tables.

Example:

```
PTERM ss19,lterm=ltdf1900,pronam=vr,ptype=t9755,TERMN=X1
```

Treatment of K Keys and F Keys

The Natural UTM Interface supports the function keys K1, K2, K3, K4, F1, F2, F3, F4 and F5 (for P keys). The function key which has been pressed can be identified by means of the UTM return code, which must be defined using the SFUNC statement of KDCDEF:

```
SFUNC K1,RET=26Z
SFUNC K2,RET=27Z
SFUNC K3,RET=28Z
SFUNC K4,RET=29Z
SFUNC F1,RET=21Z
SFUNC F2,RET=22Z
SFUNC F3,RET=23Z
SFUNC F4,RET=24Z
SFUNC F5,RET=25Z
SFUNC nn,RET=nnZ (for the PRKEY; see the keyword parameter PRKEY)
```

Using other function keys or using valid function keys that have not been defined in KDCDEF results in an error message.

Support of IBM Type 3270 Terminals

In an appropriate system configuration of IBM SNA (VTAM) and Siemens TRANSDATA DC 3270-type terminals are supported by the Natural UTM Interface.

This means that Siemens terminals as well as 3270 terminals can be connected to a Natural UTM application. Natural adjusts screen output to the specific terminal type used. 3270-type terminals have to be defined as such to KDCDEF in the PTERM command (see Siemens UTM documentations).

For the support of Siemens and IBM function keys, the SFUNC statements of KDCDEF have to be defined as follows:

Siemens Key	IBM Key	UTM Return Code
F1	PF1	21Z
F2	PF2	22Z
F3	PF3	23Z
F4	PF4	24Z
F5	PF5	25Z
K1	PA1	26Z
K2	PA2	27Z
K3	PF6 + PF13	28Z
K4	PF7 + PF14	29Z
K5	PF8 + PF15	30Z
K6	PF9 + PF16	31Z
K7	PF10 + PF17	32Z
K8	PF11 + PF18	33Z
K9	PF12 + PF19	34Z
K10	PF20	35Z
K11	PF21	36Z
K12	PF22	37Z
K13	PF23	38Z
K14	PF24	39Z

Support of TTY Terminals

For terminals which are to be used in TTY mode, the TERMN operand of the PTERM command must be set to TERMN=X9.

The following restrictions apply to TTY mode:

- Asynchronous transaction processing is not supported.
- MULTI-PASS is not supported.

UTM DC-Transaction Exit Routine NUERROR

A UTM DC-transaction exit routine is defined in the front-end part of the Natural UTM Interface. This routine is called at the beginning of a DC transaction, when a DC transaction is restarted, at normal termination and at abnormal termination (PEND ER). The user exit UVGEXIT can be used in any of these circumstances.

In the case of abnormal termination, the affected user is deleted from the internal terminal control table, the Natural recovery procedures are executed and the user's user area is released from the swap pool directory if necessary.

The DC-transaction exit routine NUERROR must be defined in the adequate parameters of KDCDEF for the front-end part of the Natural UTM Interface (generation of KDCROOT); see also Generating KDCROOT.

UTM Startup Function

If the user exit STARTEX (default value of keyword parameter STRTALL) is to be used, "EXIT PROGRAM=NUSTART,USAGE=START" must be defined in the KDCDEF parameter for the front-end part of the Natural UTM Interface.

One of the effects of this is that the task initialization routines (allocation of common memory pools, loading Natural, etc.) are activated immediately following the start of each UTM task. Errors that occur are output on the console and all users are sent an appropriate message; if SYSLST=YES (see keyword parameter SYSLST), errors are also output to SYSLST.

If the UTM startup function is not used, the UTM task(s) are not initialized until they are activated when a user logs on. If an error occurs under these circumstances, the error message is sent to the terminal that caused the error. All other users are given an appropriate message when they try to log-on to the application.

UTM Shutdown Function

If the user exits SHUTEX1 and/or SHUTEX2 (default values of keyword parameters SHUTALL and SHUTLST) are to be used, "EXIT PROGRAM=NUSTART,USAGE=SHUT" must be defined in the KDCDEF parameters (KDCROOT) for the front-end part of the Natural UTM Interface. The statistics of the Natural UTM Interface are output when the last UTM task terminates.

If the UTM shutdown function is not used, the user exits defined with SHUTALL and SHUTLST cannot be used and the statistics are not available.

The statistics that are collected and output by the Natural UTM Interface are:

```

MAIN DIRECTORY IS RESIDENT, N O T P A G E A B L E
-----
INITIALIZED WITH CONTROL DATA FROM NAT SYSTEM FILE
TOTAL SIZE OF SWAP POOL IN KB :    32760
SIZE OF MAIN DIRECTORY IN KB :      2
TOTAL NO. OF SWAP POOL THREADS:    209
TOTAL NO. OF LOGICAL SWP(S) :      10

LAST STATUS OF THE SWAP POOL STATISTICS
+-----+
I      Natural USER THREADS WITH LENGTH      I
I      GREATER                                LOWER      I
I      152 KB                                  I
+-----+
I + 2 KB:          0 I - 2 KB:          0 I
I + 4 KB:          0 I - 4 KB:          17 I
I + 6 KB:          0 I - 6 KB:          0 I
I + 8 KB:          0 I - 8 KB:          1 I
I + 10 KB:         0 I - 10 KB:         0 I
I + 12 KB:         0 I - 12 KB:         0 I
I + 14 KB:         0 I - 14 KB:         0 I
I + 16 KB:         0 I - 16 KB:         0 I
I + 18 KB:         0 I - 18 KB:         0 I
I + NN KB:         0 I - NN KB:         0 I
I AVER.LNG.NN:     0 KB I AVER.LNG.NN:   0 KB I
+-----+

LOGICAL SWP NO. 01                LOGICAL SWP NO. 02
-----
LOGICAL SWP SIZE IN KB: 2402      LOGICAL SWP SIZE IN KB: 2690
DIRECTORY SIZE IN KB :    2      DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB :  120     SWP THREAD SIZE IN KB :  128
NO. OF SWP ENTRIES :    20      NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :      0      MAX. USED ENTRIES :      0
NO. OF SWP GUESTS :      0      NO. OF SWP GUESTS :      0
NO. SUCCESSFUL LOCATES:  0      NO. SUCCESSFUL LOCATES:  0
NO. FAILED LOCATES :    0      NO. FAILED LOCATES :    0

LOGICAL SWP NO. 03                LOGICAL SWP NO. 04
-----
LOGICAL SWP SIZE IN KB: 2858      LOGICAL SWP SIZE IN KB: 3026
DIRECTORY SIZE IN KB :    2      DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB :  136     SWP THREAD SIZE IN KB :  144
NO. OF SWP ENTRIES :    21      NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :      0      MAX. USED ENTRIES :      1
NO. OF SWP GUESTS :      0      NO. OF SWP GUESTS :      0
NO. SUCCESSFUL LOCATES:  0      NO. SUCCESSFUL LOCATES:  1
NO. FAILED LOCATES :    0      NO. FAILED LOCATES :    0
    
```

```

LOGICAL SWP NO. 05
-----
LOGICAL SWP SIZE IN KB: 3194
DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB : 152
NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :     1
NO. OF SWP GUESTS :     0
NO. SUCCESSFUL LOCATES: 17
NO. FAILED LOCATES :    0

LOGICAL SWP NO. 06
-----
LOGICAL SWP SIZE IN KB: 3362
DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB : 160
NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :     0
NO. OF SWP GUESTS :     0
NO. SUCCESSFUL LOCATES:  0
NO. FAILED LOCATES :    0

LOGICAL SWP NO. 07
-----
LOGICAL SWP SIZE IN KB: 3530
DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB : 168
NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :     0
NO. OF SWP GUESTS :     0
NO. SUCCESSFUL LOCATES:  0
NO. FAILED LOCATES :    0

LOGICAL SWP NO. 08
-----
LOGICAL SWP SIZE IN KB: 3698
DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB : 176
NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :     0
NO. OF SWP GUESTS :     0
NO. SUCCESSFUL LOCATES:  0
NO. FAILED LOCATES :    0

LOGICAL SWP NO. 09
-----
LOGICAL SWP SIZE IN KB: 3866
DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB : 184
NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :     0
NO. OF SWP GUESTS :     0
NO. SUCCESSFUL LOCATES:  0
NO. FAILED LOCATES :    0

LOGICAL SWP NO. 10
-----
LOGICAL SWP SIZE IN KB: 4118
DIRECTORY SIZE IN KB :    2
SWP THREAD SIZE IN KB : 196
NO. OF SWP ENTRIES :    21
MAX. USED ENTRIES :     0
NO. OF SWP GUESTS :     0
NO. SUCCESSFUL LOCATES:  0
NO. FAILED LOCATES :    0
    
```

 USAGE STATISTICS OF SWAP POOL AND NATURAL USER THREADS

SWP NO.	SLOT NO. LNG KB	NAT. THREADS	%	DIAGRAM
01	120	0	0,0	
02	128	0	0,0	
03	136	0	0,0	
04	144	1	5,5	***
05	152	17	94,4	*****
06	160	0	0,0	
07	168	0	0,0	
08	176	0	0,0	
09	184	0	0,0	
10	196	0	0,0	
DESERTERS:		0	0,0	

```

M A I N   D I R E C T O R Y   S T A T I S T I C S   A R E A
NATSHARE: SWAP POOL START DATE           02-07-17
NATSHARE: SWAP POOL START TIME           11:52:14
NATSHARE: TOTAL NUMBER OF SWP SYNC. WAITS: 0
NATSHARE: TOTAL NUMBER OF ASYN. WRITE WAITS: 0
NATSHARE: TOTAL NUMBER OF DIALOGUE STEPS: 19
NATSHARE: TOTAL NUMBER OF SWAPS:         0
NATSHARE: TOTAL NUMBER OF WRITES TO ROLL FILE: 0
NATSHARE: TOTAL NUMBER OF SYNCHRONOUS WRITES: 0
NATSHARE: MAX NUMBER OF USER:           1
NATSHARE: MAX NUMBER OF DIALOGUES WITHOUT SWAPS: 19
    
```

```
NATSHARE: NUMBER OF SWAP POOL REORGANIZATION:      0
NATSHARE: NUMBER OF SWAP POOL REPAIR:              0
NATSHARE: NUMBER OF ABNORMAL ENDED SESSIONS:       0
NATSHARE: MAX. COMPR. L' OF NAT USER THREAD IN KB: 148
NATSHARE: REAL MAXSIZE NEEDED FROM Natural IN KB:  386
```

When Swap Pool Data Space is generated:

```
NATSHARE: D A T A   S P A C E   ( D S F ) S T A T I S T I C S
NATSHARE: NO. OF ESA DATA SPACE SLOTS:             30
NATSHARE: NO. OF WRITES INTO DATA SPACE:           0
NATSHARE: NO. OF FAILED WRITES (DATA SPACE FULL):  0
NATSHARE: NO. OF SUCCESSFUL DSP THREAD LOCATES:     0
NATSHARE: NO. OF FAILED DSP THREAD LOCATES:         0

NATSHARE: SWP STATISTICS PRINT DATE:                02-07-17
NATSHARE: SWP STATISTICS PRINT TIME:                11:52:14
```

Natural under UTM - Macro Keyword Parameters

This part of the Natural UTM Interface documentation covers the following topics:

- NATUTM Macro Keyword Parameters
- NATUTM Macro Entries
- NURENT Macro Keyword Parameters

Installation - refer to Installing the Natural UTM Interface (in the Natural Installation Guide for Mainframes).

Notation *vrs* or *vr* -> **Notation *vrs* or *vr***: If used in the following document, the notation *vrs* or *vr* stands for the relevant *version*, *release*, *system maintenance level* numbers.

NATUTM Macro Keyword Parameters

The following parameters are available:

ADACALL | ADACOM | ADAPRI | ADAUTM | APPLNAM | APRISTD | ASAPPLI | ASYNTAC | BADTAC | CDYNAM | CLRKEY | CURPRO | ICONTRL | INITPRG | KB | KBSAVE | KBUSEXT | LFH | LINK | LINK2/LINK3/LINK4 | LOFFMAP | NATMON | NUAADDR | NUCNAME | PARMOD | PENDPR | PFK | PRKEY | REFRKEY | ROLLACC | ROLLTSZ | RSTCNT | RSTWARM | SCRNOPT | SHUTALL | SHUTLST | SPOOL | STRTALL | STRTFST | SVDYPRM | SWAMODE | SWDPAGE | SWPUSID | SYAPPLI | SYNTAC | SYSLST | TACEND | TCLA1 | TCLA2, TCLA3, TCLA4 | TCLS1 | TCLS2, TCLS3, TCLS4 | TERMTAB | TID | TRACE | TTYLS | TTYPS | UMODE

ADACALL - Access to Adabas

This parameter defines an entry in the Natural UTM Interface for the subroutine ADACALL. This subroutine must be called each time a non-Natural program accesses Adabas. ADACALL generates a valid Adabas user ID and subsequently invokes the Adabas interface module ADALNN. Possible values are:

ADACALL= <i>name</i>	<i>name</i> of the entry.
ADACALL=NO	Subroutine ADACALL is not generated. This is the default value.

ADACOM - Adabas Link Module Usage

This parameter determines which Adabas link module is to be used.

ADACOM=ADABAS	The modules ADAUSER and SSFB2C are linked to the front-end part (Adabas Version 7.1 and higher).
ADACOM=ADALINK	The modules ADALNK, ADAL2P and SSFB2C are linked to the front-end part (Adabas Version 7.1 and higher).
ADACOM=,	The module ADALNN is linked to the front-end part (Adabas versions lower than 7.1). This is the default value.

ADAPRI - Activation of Adabas Priority Control for UTM Application

ADAPRI=YES	Activates Adabas priority control for a Natural UTM application.
ADAPRI=NO	The Adabas priority for all UTM transactions is the same. This is the default value.

See Adabas Priority Control for details.

ADAUTM - Synchronization of Async UTM/Adabas Transactions

This parameter enables you to realize synchronized processing and coordinated restart of asynchronous transactions between UTM and Adabas. This requires that the module "ADAUTM" is available; this module must be linked to the front-end part of the Natural UTM application. Possible values are:

ADAUTM=YES	Synchronized processing and coordinated restart of asynchronous transactions between UTM and Adabas are enabled. If ADAUTM=YES is specified, the ADACALL parameter must not be set to "Adabas".
ADAUTM=NO	This is the default value. Warning: Do not change the default value of this parameter without prior consultation of Software AG support.

APPLNAM - Name of Natural UTM Application

With this parameter, you specify the name of the Natural UTM application. The value of this parameter must be identical with the value of parameter APPLNAME in KDCDEF. This name is used to create a name for a task-specific SYSLST file. Possible values are:

APPLNAME= <i>name</i>	Up to 8 characters long. No default value provided.
-----------------------	---

The specified name is also used to construct a serialization marker for the initialization routine in the Natural UTM Interface; an "S" is inserted in the first free character position (for example, if APPLNAM=NATUTM, the name of the serialization marker is NATUTMS).

Furthermore, this name is used to create an Adabas user ID if TID=N is specified.

A defined character position of the operand of APPLNAM can be used for constructing the Adabas user ID; see keyword parameter TID.

APRISTD - Adabas Priority for Standard UTM TAC

APRISTD= <i>nnn</i>	Adabas priority <i>nnn</i> for the standard UTM TAC (default "NAT")
APRISTD=144	This is the default value.

This parameter can be used to define the Adabas priority *nnn* for the standard UTM TAC (default "NAT"). The APRISTD parameter is only in effect if the ADAPRI parameter is set to "YES". For individual TACs, individual priorities can be defined with the parameters TCLSn and TCLAn; see also Adabas Priority Control.

ASAPPLI - Name of Logical UTM Communications Partner

This parameter specifies the name of the logical UTM communications partner (as defined in KDCDEF) of the asynchronous UTM application. This name is only relevant in the case of asynchronous transaction processing between two UTM applications. Possible values are:

ASAPPLI= <i>name</i>	<i>name</i> specifies the name of the logical UTM communications partner.
ASAPPLI=NO	This is the default value.

If ASAPPLI=*name* is specified, the operand of the keyword parameter SYAPPLI must also be defined.

ASYN_TAC - UTM Transaction Code for Asynchronous UTM Task or Application

With this parameter you define the UTM transaction code (TAC) for the UTM task or application that runs asynchronously. Possible values are:

ASYN_TAC= <i>tac</i>	UTM TAC for the UTM task or application that runs asynchronously.
ASYN_TAC=NATAS	This is the default value.

The specified UTM TAC must be distinct from the "standard" Natural TAC and also from the TAC used for the synchronous UTM application (if asynchronous transaction processing is used between two UTM applications).

The first five characters determine the unique identifier for asynchronous UTM TACs.

BADTAC - Activation of UTM Function "BADTACS"

This parameter enables you to activate the UTM function "BADTACS", which means that in the assembled program of macro NATUTM, the startup program "AUTOTAC" is generated for undefined UTM transaction codes. Possible values are:

BADTAC=YES	Activates the UTM function "BADTACS".
BADTAC=NO	This is the default value.

BADTAC=YES requires that the following additional definitions must be supplied when defining KDCDEF and generating KDCROOT:

<pre>KDCDEF: PROGRAM AUTOTAC , COMP=ASSEMB TAC KDCBADTC , CALL=FIRST , PROGRAM=AUTOTAC , EXIT=NUERROR , TYPE=D TAC AUTOCONN , TYPE=D , PROGRAM=NATUTM , EXIT=NUERROR , CALL=BOTH</pre>
--

CDYNAM - Maximum Number of Programs to be Dynamically Loaded

This parameter specifies the maximum number of programs to be dynamically loaded (for example, COBOL or Assembler subroutines) and/or the number of programs which have been linked with the front-end part and declared with parameters LINK to LINK4. Possible values are:

CDYNAM= <i>nn</i>	<i>nn</i> defines the number of programs.
CDYNAM=15	This is the default value.

The programs to be dynamically loaded must be either in the load library specified in the Natural parameter module or in the BLSLIB(s) specified in the start job.

CLRKEY - Activation/Deactivation of CLEAR Key

This parameter activates or deactivates the CLEAR key. Possible values are:

CLRKEY=ON	Activates the CLEAR key (keys LSP and ENTER). This is the default value.
CLRKEY=OFF	Deactivates the CLEAR key, which means that after pressing CLEAR, the entire last Natural screen is displayed again.

CURPRO - Cursor Positioning to Protected Field

This parameter controls whether the cursor can be positioned to a protected field. Possible values are:

CURPRO=ON	The cursor cannot be positioned to a protected field. This is the default value.
CURPRO=OFF	The cursor can also be placed in a protected field (for example, for field-specific help functions).

ICONTRL - UTM Input Exit for Messages in Minus Format

This parameter allows you to generate an UTM input exit for messages in minus ("-") format; that is, messages from a Natural screen. Such an input exit controls the allowed (or not-allowed) user KDC commands. Possible values are:

ICONTRL=(YES,KDCxxx(,KDCxxx,...))	Any KDC command not allowed must be defined with this parameter by specifying YES and the name of the KDC command. See examples below.
ICONTRL=(YES)	
ICONTRL=(NO)	This is the default value.

Examples:

ICONTRL=(NO)	This example does not generate an input exit and allows all UTM commands.
ICONTRL=(YES)	This example generates an input exit with the name ICONTRL and prohibits usage of all UTM commands.
ICONTRL=(YES,KDCOUT,KDCOFF)	This example generates an input exit with the name ICONTRL and prohibits usage of the commands KDCOUT and KDCOFF.

If YES is specified as first operand, the generated input exit must be defined in KDCDEF and KDCROOT as follows:

```
EXIT PROGRAM=ICONTRL , USAGE=( INPUT , USERFORM )
PROGRAM ICONTRL , COMP=ASSEMB
```

INITPRG - Value for Natural Variable *INIT-PROGRAM

This parameter defines the value for the Natural variable *INIT-PROGRAM. Possible values are:

INITPRG=APPLNAM	The Natural variable *INIT-PROGRAM contains the value of the keyword parameter APPLNAM. This is the default value.
INITPRG=KCTACVG	The Natural variable *INIT-PROGRAM contains the value of the UTM KB field KCTACVG (UTM start TAC).

KB - Pass KB Address as First Parameter

This parameter specifies whether the address of the UTM communication area (Kommunikationsbereich, KB) is passed as the first parameter address each time Natural calls a non-Natural program. This has been taken account of in the subroutines and utility programs of the Natural UTM Interface. Possible values are:

KB=YES	The address of the UTM communication area (KB) is passed as the first parameter address each time Natural calls a non-Natural program.
KB=NO	This is the default value.

KBSAVE - Saving of UTM KB via SPUT

This parameter specifies whether the UTM KB will be saved via SPUT or not. Possible values are:

KBSAVE=YES	The UTM KB will be saved via SPUT, starting from the end of the KB header plus twelve bytes. This information will be saved in the LSSB before a PEND PR(ogram) is executed for a user-specific partial UTM program.
KBSAVE=NO	The UTM KB will not be saved. This is the default value.

To be able to use this parameter, you must set the following KDCDEF definition:

```
MAX LSSBS=1
```

If the user-specific partial UTM program resumes, the original communication area will be refreshed via SGET. This allows the partial UTM program to use the KB from the end of the UTM communication area header plus twelve bytes. Therefore, the program must not destroy these twelve bytes. If a KB user extension is defined, this area will not be saved.

KBUSEXT - Length of UTM KB User Extension

This parameter specifies the length of a UTM KB (Kommunikationsbereich) user extension. Possible values are:

KBUSEXT=nnnnn	nnnnn specifies the length of a UTM KB user extension. The maximum length allowed is 30720 bytes.
KBUSEXT=0	This is the default value.

Length and address of a user extension are stored in the KB:

USEREXTL DS	H	<i>length in bytes</i>
USEREXTA DS	F	<i>address</i>

For more information, see the DSECT macro CMBS2TP.

LFH - Use of Adabas LFH

This parameter specifies that the Adabas large file handler (LFH) is to be used. Possible values are:

LFH=YES	Specifies that you are using the Adabas LFH .
LFH=NO	This is the default value.

If you specify YES, you also must define the buffer size for the Adabas LFH in the Natural parameter module (parameter VSIZE).

LINK - Programs and Modules Called from Natural

This parameter enables you to specify the names of programs and modules that are called from Natural programs and linked with the non-reentrant part. Possible values are:

LINK= <i>name</i> (<i>name,name,...</i>)	<p>The <i>names</i> of programs and modules that are called from Natural programs and linked with the non-reentrant part must be specified in the operand of this parameter. Conversely, the programs and modules whose names are specified must be linked with the non-reentrant part, otherwise the application is put into status SYSTEMERROR and all users are rejected with an error message.</p> <p>Default value: none</p>
---	---

A "TABLE" macro call is performed for the specified programs and modules, which enters their load addresses into the dynamic loader's link table. It is therefore not necessary to dynamically load these programs when they are called by Natural programs.

Example:

```
LINK=PROG1
LINK=( PROG1 , PROG2 , MODUL111 )
```

LINK2/LINK3/LINK4 - Extensions of Keyword Parameter LINK

These parameters are an extension of the keyword parameter LINK. Possible values are:

LINK2/LINK3/LINK4= <i>name</i> (<i>name,name,...</i>)	<p>The syntax is in analogy to that of LINK. See examples below.</p> <p>Default value: none</p>
--	---

Since an operand definition must not be longer than 127 characters (including parentheses), the parameters LINK2 to LINK4 are provided for cases where the operand of parameter LINK would be too long.

Examples:

```
NATUTM LINK=( PROG1 , PROG2 , . . . ) ,
LINK2=( PROG54 , . . . )
NATUTM LINK=( PROG1 , PROG2 , PROG3 , PROG4 )
```

LOFFMAP - Format Name for Logoff Message

With this parameter, a format name for the logoff message can be specified. Possible values are:

LOFFMAP=' -END'	<p>The message defined in the format exit module FREXIT is output:</p> <pre>NAT9994 - YOUR SESSION WAS SUCCESSFULLY FINISHED. PLEASE GIVE "KDCOFF" (LEAVE THE APPLICATION) OR "UTM-TAC" .</pre> <p>The message is output in the language specified by parameter UMODE; if required, it can be modified in the program FREXIT.</p> <p>This is the default value.</p>
LOFFMAP=' '	<p>The following message is output in line mode:</p> <pre>NAT9994 - Natural TERMINATED NORMALLY</pre>
LOFFMAP='name' ("-" or "*" format)	<p>The user-defined message is output.</p> <p>The message is defined with "-" format in FREXIT or with "*" format with IFG and FHS.</p>
LOFFMAP='KDCOFF'	<p>An automatic "KDCOFF" is performed for the user when a FIN system command or TERMINATE statement is executed.</p>

In any case, the operand specified with the LOFFMAP parameter is used as the format name for UTM. The operand is therefore restricted to a maximum of 8 characters.

NATMON - Automatic Activation of Natural Monitor during Application Startup

This parameter specifies whether the Natural monitor is activated automatically during application startup or not. Possible values are:

NATMON=ON	The Natural monitor is activated automatically during application startup.
NATMON=OFF	The Natural monitor is not activated automatically during application startup. This is the default value.

NUAADDR - Natural User Thread Address

With this parameter, you specify a Natural user thread address. The following happens if you specify a particular value:

NUAADDR=, (comma)	Comma means no value. The Natural user thread will be allocated in the next free address below the 16-MB line. This is the default value.
NUAADDR=XXXXX	The Natural user thread will be allocated on the hexadecimal address in the class 6 memory below the 16-MB line. This address must be aligned to the 4-KB segment limit. The result of address plus Natural user thread's length in bytes (MAXSIZE) must not be greater than address X'DF0000'. The highest possible address is X'DEFFFF'.
NUAADDR=ABOVE	The Natural user thread will be allocated above the 16-MB line.
NUAADDR=(ABOVE,NNNNN)	The Natural user thread will be allocated above the 16-megabyte line where NNNNN denotes the decimal number of megabytes above the 16-MB line.

Examples:

NUAADDR=ABOVE	The Natural user thread will be allocated in the next free address above the 16-MB line.
NUAADDR=(ABOVE,258)	The hexadecimal address of the Natural user thread is X'10200000' (above the 16-MB line).
NUAADDR=6E000I	The hexadecimal address of the Natural user thread is X'6E000' (below the 16-MB line).

When the Natural user thread is allocated above the 16-MB line, the asynchronous write buffer and the thread for asynchronous transactions will also be allocated above the 16-MB line. In this case, the 31-bit address mode will not be switched back to 24-bit address mode before a 3GL program is called. This means the called 3GL program must be able to run in 31-bit address mode.

NUCNAME - Name of Bounded Reentrant Natural Module

This parameter specifies the name of the bounded, reentrant Natural module. Possible values are:

NUCNAME= <i>name</i>	<i>name</i> of the bounded, reentrant Natural module. Default value: none
----------------------	--

You must use the name of the bounded, reentrant Natural module for the Natural pool and load information in macro ADDON (macro ADDON assembles BS2STUB) and for program CMPSTART when a shared nucleus is to be used.

PARMOD - Generation of Front-End and Reentrant Parts

This parameter applies to the generation of both the front-end and reentrant parts. Possible values are:

PARMOD= <i>nn,loc</i> <i>nn</i> : 24/31 <i>loc</i> : BELOW/ABOVE	The first value of this parameter (<i>nn</i>) is used to define an addressing mode (24-bit or 31-bit mode) for the Natural UTM application. The second value of this parameter (<i>loc</i>) is used to define the front part location of the Natural UTM application.
PARMOD=(31,ABOVE)	This is the default value.

If you load the front part of the application above 16 MB, this must be defined in the front part's link procedure as follows:

```
LOADPT=*XS or
LOADPT=X'address'
```

Example:

```
/EXEC TSOLINK
  PROG NATvrs, FILENAM=NATvrs, LOADPT=*XS, ...
  TRAITS RMODE=ANY, AMODE=31
  INCLUDE...
/* PARMOD=(nn, loc) MUST ALSO BE DEFINED FOR ASSEMBLING MACRO NURENT, WHICH
/* BELONGS TO THE REENTRANT PART OF Natural/UTM; OPERANDS MUST BE IDENTICAL FOR
/* THE FRONT-END AND REENTRANT PARTS.
```

PENDPR - Define UTM TAC for "PEND PR"

This parameter defines a UTM TAC for a "PEND PR". Possible values are:

PENDPR='zzzzzzzz'	zzzzzzzz (maximum 8 characters) defines the UTM TAC.
PENDPR=' '	This is the default value (no TAC for PEND PR) .

When PENDPR='zzzzzzzz' is specified, a "PEND PR(OGRAM)" is executed instead of a "PEND FI(NISH)" when the FIN system command is entered or a TERMINATE statement is executed or the PEND PR function key is pressed. The UTM partial program that has been associated with the specified UTM TAC is started after the PEND PR.

PFK - Function Key Modes

This parameter is used to set one of the following function-key modes:

PFK=(KN,y)	The literals "%K1" to "%K20" and send-key code "DÜ1" are loaded to the function keys.
PFK=(KO,y)	The literals "01" to "20" and send-key code "F5" are loaded to the function keys.
PFK=(KS,y)	The literals "A" to "T" and send-key code "F5" are loaded to the function keys; in addition, with every output message a dummy field is generated at the last two positions of the screen, which is used to receive and pass the key value.
PFK=OFF	No function key mode is generated.
PFK=KS,L	This is the default value.

Where y can be:

- "L" - function keys are loaded, or
- "N" - function keys are not loaded.

PRKEY - UTM Return Code for Function Key

This parameter is used to define a UTM return code for a function key (F1 to F5 or K1 to K14). Possible values are:

PRKEY=nmZ	Possible values are 20Z to 39Z.
PRKEY=35Z	Default value for K10 (keys ESC + ">").

Whenever a function key defined with this parameter is activated in the Natural dialogue, the Natural session is suspended and if a UTM TAC for another UTM partial program is available, a PEND PR(OGRAM) is executed.

This UTM TAC can be defined in several ways:

- with the Natural profile parameter PROGRAM=*tac*,
- with the keyword parameter PENDPR=*tac*,
- with the utility program TACSWTCH.

On return from the called UTM partial program via the PEND PR(OGRAM) to the Natural UTM Interface, the Natural session is continued at the point where it has been suspended.

The same return code as specified with the PRKEY parameter must also be defined with an SFUNC statement in KDCDEF.

REFRKEY - Definition of UTM Function Key

This parameter can be used to define a UTM function key. Possible values are:

REFRKEY= <i>nmZ</i>	Possible values for <i>nm</i> are in the range from 26 to 39 (K1 to K14).
REFRKEY=NO	No UTM function key defined.
REFRKEY=39Z	Default value for K14 (keys ESC + ":").

If the defined function key is pressed, the last full Natural screen is refreshed. Thus it is possible to continue the dialog with Natural after the screen has been overwritten by messages from the operator or the operating system. The send key code is not passed to the Natural application. The interface sets the Natural key code to "ENTER".

ROLLACC - Access Method for Natural Roll File

This parameter defines the access method for the Natural roll file. Possible values are:

ROLLACC=UPAM-SY	The access method for the Natural roll file is UPAM with synchronous roll file I/Os. This access method is not allowed with global swap pools.
ROLLACC=UPAM-AS	The access method for the Natural roll file is UPAM with P1-eventing for asynchronous writes. This is the default value.
ROLLACC=(UPAM-AS,PAMWAIT)	The NATURAL UTM interface waits with a VPASS SVC from the completed asynchronous write before a PEND RE is executed. This option is needed because a UTM task which is inactive (P2 wait) cannot be posted via P1-eventing. Instead, the user session must be terminated with the error message "Timeout for asynchronous write."
ROLLACC=FASTPAM	The access method for the Natural roll file is FASTPAM with Forward Eventing for asynchronous writes (high performance). See prerequisites described below.

Prerequisites for ROLLACC=FASTPAM

To use the FASTPAM option, the following prerequisites apply:

BS2000/OSD Version 1.0 or above.

- Parameter TERMTAB must be defined as SWP.
- The class II definition in the batch job for starting the resident FASTPAM environment and the FASTPAM I/O pool must be:
/EXEC NATUTM,CLASSII=(*nnn*,*yy*)
- The FASTPAM authorization in the user catalog must be:
/SHOW-USER-ATTRIBUTES
FIELD: DMS-TUNING-RESOURCES=*EXCLUSIVE
/*OR ALTERNATIVELY:
/MODIFY-USER
FIELD: DMS-TUNING-RESOURCES=EXCLUSIVE-USE
- The BIAS for the BS2000/OSD operating system must be defined as follows:
/MODIFY-SYSTEM-BIAS MAX-RESIDENT-PAGES=*nnn*

To calculate the necessary number of resident core pages, use the following formula (ignore all rest values):

$ROLLTSZ + 3 / 4 * 2$	= N1 (FASTPAM I/O areas)
$ROLLTSZ + 31 / 32 * 36 + 4095 / 4096 * 2$	= N2 (FASTPAM access lists)

$N1 + N2$ = number of resident pages for one Natural/UTM task

ROLLTSZ - Maximum Roll Thread Size

This parameter determines the maximum roll thread size *nnn* (in KB); that is, the maximum size of a compressed user thread on the Natural roll file. Possible values are:

ROLLTSZ= <i>nnn</i>	<i>nnn</i> must be a multiple of 4 (roll file block size).
ROLLTSZ=160	Default value: 160 KB

If ROLLACC=UPAM-AS, valid values for ROLLTSZ are 4 to 1600 (KB).

If ROLLACC=UPAM-SY or ROLLACC=FASTPAM, valid values for ROLLTSZ are 4 to 3200 (KB).

To calculate the size of the Natural roll file, use the following formula:

$$ROLLTSZ / 2 * \text{maximum number of users} = nnn$$

nnn is the number of PAM pages for the Natural roll file.

As user threads are generally written to the roll file in compressed form, an optimum roll thread size contributes considerably to saving disc storage.

The optimum value for *nnn* can be ascertained with the Natural Swap Pool Statistics; see the SYSLST parameter.

RSTCNT - Control of Restart Situations

This parameter can be used to control restart situations in which the "lifetime" of a user results from an old Natural/UTM session. Possible values are:

RSTCNT=YES	In such a restart situation a message is displayed to the user and the UTM task is finished with "PEND FI(NISH)"; the user must restart his/her UTM task by entering the UTM TAC.
RSTCNT=NO	In such a restart situation the Natural session is newly initialized without a message being displayed. This is the default value.

RSTWARM - Control of Restart Situations

This parameter can be used to control restart situations. Possible values are:

RSTWARM=YES	There will be a warm start of a Natural session if there is a UTM restart situation. The last terminal screen will be displayed, prerequisite for this function is a global Natural swap pool.
RSTWARM=NO	There will be a restart of a Natural session if there is a UTM restart situation. This is the default value.

SCRNOPT - Terminal Types with Deactivated Natural Screen Optimization

This parameter can be used to define (one or two) terminal types for which Natural screen optimization is to be de-activated. Possible values are:

SCRNOPT=(yy=zz)	yy must be a valid terminal name TERMN as defined in KDCDEF. zz is a synonym for yy. For terminal types defined in KDCDEF with TERMN=zz, screen optimization is then de-activated.
SCRNOPT=(yy=zz,yy=zz)	Same as above, but two terminals defined.
SCRNOPT=NO	Screen optimization is active for all terminal types. This is the default value.

Example:

```
SCRNOPT=( FL=Z9 )
```

"FL" is a valid TERMN name for IBM 3270-type terminals; "Z9" is a synonym for 3270-type terminals. This example would deactivate screen optimization for those 3270-type terminals which are defined as TERMN=Z9 in KDCDEF.

SHUTALL - Name of User Exit

With this parameter, you can specify the *name* of a user exit. Possible values are:

SHUTALL= <i>name</i>	Specifies the name of a user exit.
SHUTALL=SHUTEX1	This is the default value.

This user exit is invoked by the Natural UTM interface whenever a UTM task is terminated with "KDCSHUT", provided that the UTM SHUTDOWN function has been defined in KDCDEF.

SHUTLST - Name of User Exit

With this parameter, you can specify the *name* of a user exit. Possible values are:

SHUTLST= <i>name</i>	Specifies the name of a user exit.
SHUTLST=SHUTEX2	This is the default value.

This user exit is invoked by Natural/UTM when the last UTM task is terminated with "KDCSHUT", provided that the UTM SHUTDOWN function has been defined in KDCROOT.

SPOOL - Automatic Start and Termination of Printer Task

This parameter enables you to specify a spooling system. Possible values are:

SPOOL=(NATSPOOL,' <i>enter-parms</i> ', <i>n</i>)	For use with NATSPOOL (Natural Advanced Facilities), see Using NATSPOOL.
SPOOL=REPRO-2000	For use with a remote spooling system, see Using REPRO-2000 Remote Spooling System.
SPOOL=RMSPOOL	For use with your own user exit program, see Using RMSPOOL User Exit.

Default value: None.

Using NATSPOOL

When using NATSPOOL (Natural Advanced Facilities), the SPOOL parameter can be used to indicate that the printer task(s) required by NATSPOOL are to be started up automatically by means of ENTER calls whenever the Natural UTM application is started, and terminated whenever the application is shut down. In this case, the operands of the keyword parameter must be:

```
SPOOL=(NATSPOOL,'enter-parm',n)
```

where '*enter-parms*' are the parameters for the ENTER call (in apostrophes) and *n* is the number of printer tasks to be started (in the range 1 to 30).

Example:

The following ENTER job is to be automatically started and terminated. The file name is "AF.E.PRINT":

```
/LOGON
/OPTION MSG=FHL
/SYSFILE FILE=SYSLST
/EXEC NAFPTTSK
/LOGOFF
```

Operand definition for the keyword parameter SPOOL:

```
SPOOL=(NATSPOOL,'AF.E.PRINT',TIME=999',2)
```

In this example, "NATSPOOL" is the name of the Natural spooling system; "AF.E.PRINT" is the file name of the ENTER job to be started and terminated; "TIME=999" is an additional, optional parameter for the ENTER call (see the description of the BS2000/OSD ENTER macro); and "2" means that two NATSPOOL printer tasks are to be started/terminated.

The second suboperand can contain any valid operands (enclosed in apostrophes) for the ENTER macro call.

The operand of keyword parameter SPOOL in macro NURENT must be "NATSPPOOL".

NATSPPOOL Processing Logic

The specified number of NATSPPOOL printer tasks according to the operand definition in the keyword parameter SPOOL is started when the application is started up. Interprocess communication is then used to check that at least one printer task is running. If this condition is not satisfied, the application is set to status SYSTEMERROR, an error message is output on the console and users who attempt to logon are rejected with the message:

```
NUI0036 - SYSTEMERROR ... PLEASE GIVE KDCOFF
```

For more information on this system error, see NUI0036.

Using REPRO-2000 Remote Spooling System

If a remote spooling system is used (for example, TD-SPOOL or REPRO-2000), set SPOOL=REPRO-2000 in the macros NATUTM and NURENT. This function is not supported by Software AG.

The logic used by Natural offline reports must be considered when implementing the interface module for a remote spooling system (see macro NURENT, label CMWHC). When an offline report is activated, Natural transfers output a record at a time. The logic for sending and accepting print records, the layout of the print record, etc., are in macro NURENT, subroutine CMWHC.

Using RMSPOOL User Exit

If you use your own user exit program named "RMSPOOL" as remote spooling interface, set SPOOL=RMSPOOL in the macros NATUTM and NURENT. See User Exits for details on the user exit "RMSPOOL".

STRTALL - Name of User Exit for All UTM Tasks

With this parameter, you can specify the *name* of a user exit. This user exit is invoked by Natural/UTM whenever a UTM task is started. Possible values are:

STRTALL= <i>name</i>	Specifies the <i>name</i> of a user exit.
STRTALL=STARTEX	This is the default setting.

STRTFST - Name of User Exit for First UTM Task

With this parameter, you can specify the *name* of a user exit. This user exit is invoked by Natural/UTM when the first UTM task is started, provided that the UTM STARTUP function has been defined in KDCDEF. Possible values are:

STRTFST= <i>name</i>	Specifies the <i>name</i> of a user exit.
STRTFST=STAPPLX	This is the default setting.

SVDYPRM - Save Area Length for Dynamic Natural Parameters

This parameter determines the length in bytes of a save area for dynamic Natural parameters in the UTM KB. These parameters are used when a Natural/UTM session is restarted. Possible values are:

SVDYPRM= <i>nnnn</i>	Specifies the length in bytes of a save area for dynamic Natural parameters in the UTM KB. Possible values are 0/8...2048 (bytes).
SVDYPRM=0	This is the default value.

SWAMODE - Switching from 31 to 24-Bit Address Mode

This parameter determines whether a 31-bit address mode is switched to 24-bit mode or not before a PEND PR(ogram) is executed. What you must set depends on whether the partial UTM program can run in 31-bit address mode (NO) or not (YES). Possible values are:

SWAMODE=YES	31-bit address mode is switched to 24-bit mode.
SWAMODE=NO	This is the default value.

SWDPAGE - Pageability of Swap Pool Main Directory

This parameter determines whether the swap pool main directory is pageable or not. Possible values are:

SWDPAGE=NO	Specifies that the swap pool main directory is not pageable.
SWDPAGE=YES	This is the default value.

A swap pool directory that is not pageable improves performance considerably. In that case, the BS2000/OSD macro CSTAT will be used to declare the swap pool directory as not pageable. To be able to specify SWDPAGE=NO, you must define the maximum and minimum of resident core pages in the startup job.

Example: /EXEC E.NATvrs,CLASSII=(4,2)

For more information, see the description of BS2000/OSD macro CSTAT or the description of BS2000/OSD command EXECUTE, operand CLASSII or, when SDF is used, the description of BS2000/OSD command START-PROGRAM, operand RESIDENT-PAGES=PARAMETERS....

If the call to macro CSTAT fails, the application is still able to run.

SWPUSID - Swap Pool User Identification

This parameter determines the swap pool user identification. Possible values are:

SWPUSID=KCLOGTER	This is the UTM KB's logical terminal name. This is the default value.
SWPUSID=KCBENID	This is the UTM KB's user name.
SWPUSID=INTERNID	This is the internal terminal ID (serial number).

SYAPPLI - Name of Logical UTM Communications Partner

With this parameter, you can specify the *name* of the logical UTM communications partner (as defined in KDCDEF) of the synchronous UTM application. Possible values are:

SYAPPLI= <i>name</i>	The operand of the keyword parameter ASAPPLI must also be defined.
SYAPPLI=NO	This is the default value.

The operand is only significant in the case of asynchronous transaction processing between two UTM applications.

SYNTAC - UTM TAC for Sending Messages from Async to Sync UTM Applications

This parameter defines the UTM transaction code used to send free messages for a terminal from the asynchronous to the synchronous UTM application. Possible values are:

SYNTAC= <i>tac</i>	Specifies the UTM transaction code.
SYNTAC=NATSY	This is the default value (synchronous TAC).

The UTM TAC specified in this parameter must be distinct from the "standard" Natural TAC and also from the TAC used for the asynchronous UTM application.

SYSLST - SYSLIST File Generation for UTM Task

This parameter defines whether a SYSLST file is generated for each UTM task or not. The SYSLST file contains statistics data and error information (if a UTM task ends abnormally). Possible values are:

SYSLST=YES	A SYSLST file is generated for each UTM task. This is the default value.
SYSLST=NO	No SYSLST file is generated.

The name of a SYSLST file is "LST.*name.tsn*", which is generated from the following components:

- LST - prefix,
- *name* - the value of keyword parameter APPLNAM,
- *tsn* - the 4-digit task sequence number of the UTM task.

TACEND - Action at PEND

This parameter defines the action to be taken in conjunction with the UTM operation key "PEND". Possible values are:

TACEND=KP	Each dialogue step is terminated with a "PEND KP" (KEEP). The UTM-KB is written to the page pool of KDCFILE only if no additional space in UTM CACHE storage is available. It is to be noted that no synchronized processing between UTM(s) and Adabas can be performed.
TACEND=RE	Each dialogue step is terminated with a "PEND RE" (RETURN); that is, the end of a UTM transaction. The UTM-KBs in the page pool of KDCFILE are saved with each dialogue step. This processing mode is required when a synchronized processing between UTM(s) and Adabas is to take place. This is the default value.

TCLA1 - UTM TACs for Async Transaction w. Priority Level 1

This parameter allocates UTM TACs for asynchronous transactions with priority level 1 using the UTM TACCLASS concept. A TAC table is constructed that can be accessed from Natural programs by means of the subroutine NATTAC, passing a priority level as parameter; see UTM TACCLASS Concept (Priority Control).

Possible values are:

TCLA1= <i>tac</i> TCLA1=(<i>tac,nn</i>) TCLA1=- (-, 0)	<i>nn</i> can be specified to control Adabas priority for the corresponding UTM TAC (TACCLASS); see Adabas Priority Control. Specifying TCLA1=- (note that the dash is not enclosed in apostrophes) denotes that no UTM TAC is to be allocated.
TCLA1=(NATAS1,64)	This is the default value.

TCLA2, TCLA3, TCLA4 - UTM TACs for Async Transaction w. Priority Levels 2, 3, 4

These parameters allocate UTM TACs for asynchronous transactions with priority levels 2, 3 and 4 using the UTM TACCLASS concept. Their values are used analogous to TCLA1 (see above). Possible values are:

TCLAn= <i>tac</i> TCLAn=(<i>tac,nn</i>) TCLAn=-	Analogous to TCLA1, but for priority levels $n=2, 3, 4$.
TCLA2=(NATAS2,48) TCLA3=(NATAS3,32) TCLA4=(NATAS4,16)	These are the default values.

TCLS1 - UTM TACs for Async Transaction w. Priority Level 1

This parameter allocates UTM TACs for synchronous transactions with priority level 1 using the UTM TACCLASS concept. A TAC table is constructed that can be accessed from Natural programs by means of the subroutine NATTAC, passing a priority level as parameter; see UTM TACCLASS Concept (Priority Control).

Possible values are:

TCLS1= <i>tac</i> TCLS1=(<i>tac,nn</i>) TCLS1=- (-, 0)	<i>nn</i> can be specified to control Adabas priority for the corresponding UTM TAC (TACCLASS); see Adabas Priority Control. Specifying TCLS1=- (note that the dash is not enclosed in apostrophes) denotes that no UTM TAC is to be allocated.
TCLS1=(NAT1,128)	This is the default value.

TCLS2, TCLS3, TCLS4 - UTM TACs for Async Transaction w. Priority Levels 2, 3, 4

These parameters allocate UTM TACs for asynchronous transactions with priority levels 2, 3 and 4 using the UTM TACCLASS concept. Their values are used analogous to TCLS1 (see above).

Possible values are:

TCLSn= <i>tac</i> TCLSn=(<i>tac,nn</i>) TCLSn=-	Analogous to TCLS1, but for priority levels $n=2, 3, 4$.
TCLS2=(NAT2,112) TCLS3=(NAT3,96) TCLS4=(NAT4,80)	These are the default values.

TERMTAB - Terminal Control Table for Natural Roll File Management

This parameter defines the terminal control table needed to manage the Natural roll file. Possible values are:

TERMTAB=(SWP,TERMNAME,CHECKPNT)	20 bytes long
TERMTAB=(SWP,INTERNID,CHECKPNT)	12 bytes
TERMTAB=(SWP,TERMNAME)	10 bytes
TERMTAB=(SWP,INTERNID)	2 bytes
TERMTAB=(N,TERMNAME)	10 bytes
TERMTAB=(N,INTERNID)	2 bytes
TERMTAB=(SWP,TERMNAME)	Default value.

The terminal control table is allocated either in the Natural swap pool or in the Natural roll file. It contains a header (48 bytes) and an entry for each active user or active session. Its size depends on the size of the Natural roll file, on the value of the parameter ROLLTSZ and on the length of its own entries.

The Natural UTM Interface computes the length of the terminal control table as follows:

```
Roll file pages / (ROLLTSZ / 2) = N
N * terminal control table entry length + 48 = length of the terminal control table
```

SWP	The terminal control table is allocated in the Natural swap pool.
TERMNAME	The logical terminal name will be used to identify an entry in the terminal control table.
INTERNID	The internal terminal ID (serial number) will be used to identify an entry in the terminal control table. INTERNID is two bytes long.
CHECKPNT	Is only allowed when the terminal control table is allocated in the Natural swap pool. It is necessary if terminals are defined with RESTART=NO or if a terminal pool is defined in KDCDEF. The terminal control table entry contains a checkpoint (timestamp) for the last Natural user thread that has been rolled out. A user thread in the Natural roll file should not be overwritten by a thread with a timestamp lower than the timestamp in the terminal control table entry. CHECKPNT is 10 bytes long.
N	The number of PAM pages for the terminal control table in the Natural roll file. Possible values of this operand are 1...16 (PAM pages). For each terminal, 10 bytes are needed in the terminal control table. For each session, two bytes are needed.

Examples:

```
TERMTAB=( 2 , TERMNAME )
```

The maximal number of entries in the terminal control table: $2 * 2048 - 48 / 10 = 404$

```
TERMTAB=( 1 , INTERNID )
```

The maximal number of entries in the terminal control table: $1 * 2048 - 48 / 2 = 1000$

TID - Adabas User ID Construction Method

This parameter specifies the method to be used to construct the "unique" Adabas user ID. Possible values are:

TID= <i>n</i>	The Adabas user ID is constructed from the defined (<i>n</i>) character of the operand of the keyword parameter APPLNAM (default value: N) and the last two characters of the user's first SWAPPAMKEY. <i>n</i> must be a number in the range of 1 to 8. Important: If you specify TID= <i>n</i> , the "defined character" of the value specified with the keyword parameter APPLNAM must be different from that of other Natural UTM applications if these use the same Adabas; otherwise, the uniqueness of the Adabas user IDs - and thus data consistency - cannot be guaranteed.
TID=(T, <i>n</i>)	A unique 4-byte user ID is constructed by taking characters <i>n</i> to (max. <i>n</i> +3) of the logical UTM terminal name (KCLOGTER). <i>n</i> must be a number in the range 1 - 8. The resulting character string must consist of valid characters (0 - 9 and A - F) and must be unique. See example below.
TID=(U, <i>n</i>)	The characters are taken from the UTM user ID (KCBENID), starting at the position specified by the second subparameter. The resulting character string must consist of valid characters and must be unique.
(TID=1)	This is the default value. The Adabas user ID consists of the first digit from the operand of keyword parameter APPLNAM and of the two-byte entry number in the terminal control table.

Example:

TID= (T , 4)	KCLOGTER	Adabas User ID
1st terminal	LTU9A110	X'00009A11'
2nd terminal	LTU9F110	X'00009F11'
3rd terminal	LTU9F120	X'00009F12'

If "mixed" Adabas calls occur within one Natural UTM application (that is, calls from both Natural and non-Natural programs), the Adabas user ID can be found using the "ENTRY CMTRMID" in macro NATUTM. The current Adabas user ID (4 bytes) can be found at address CMTRMID; see also the keyword parameter ADACALL.

Example:

```
EXTRN CMTRMID
.
.
L R1,CMTRMID
MVC ADAID(4),0(R1)
```

With Adabas Version 5.2:

With Adabas Version 5.2, the Adabas user IDs are constructed either from the logical UTM terminal name (KCLOGTER) or from the UTM user ID (KCBENID); this depends on the setting of the keyword parameter TID. For compatibility reasons, the second digit of the operand specified with TID must be a number from 1 to 8. For example:

```
KCLOGTER=LTU9A110    KCBENID=HARRIS

TID=(T,3) -> Adabas user ID: D9A110
TID=(U,1) -> Adabas user ID: HARRIS__
```

Note: For "mixed" Adabas calls no particular considerations have to be observed.

In the case of asynchronous transaction processing, the Adabas user ID is constructed from the "packed" TSN of the asynchronous task.

With Adabas Version 5.3 and above:

With Adabas Version 5.3 and above, for the Adabas user ID, the full terminal name (KCLOGTER) will be used when TID=(T,n) or the full user ID (KCBENID) will be used when TID=(U,n). Default is TID=((T,1).

TRACE - Trace File Number and Trace Print Record Length

With this parameter, you specify the number of a trace file and the maximal length of a trace print record. Possible values are:

TRACE=(<i>nn,ll</i>)	<i>nn</i> is the number for the SYSLST <i>nn</i> trace file. Possible range: 01...99 <i>ll</i> is the maximal length in characters of a trace print record. Possible range: 71...132
TRACE=(99,71)	This is the default value.

If any external Natural trace function is active, the trace records will be written to SYSLST*nn*. In this case, the Natural/UTM driver creates the following trace file:

Example:

```
applname.Natural.TRACE,SPACE=(90,60)
SYSFILE SYSLSTnn=applname.Natural.TRACE
/* applname is the application name
```

This file will be used by all tasks of the Natural UTM application. Before the Natural UTM application is terminated, the trace file will be closed as follows:

```
SYSFILE SYSLSTnn=(PRIMARY)
```

To activate the Natural trace functions, see the parameters ETRACE and ITRACE of the Natural parameter module.

TTYLS - Physical Line Size for TTY Devices

With this parameter you can adjust Natural's physical line length to different paper formats used with a telex machine. Possible values are:

TTYLS= <i>nn</i>	<i>nn</i> specifies the physical line size for TTY devices.
TTYLS=80	This is the default value.

TTYPS - Physical Page Size for TTY Devices

With this parameter you can adjust Natural's physical page size to different paper formats used with a telex machine. Possible values are:

TTYPS= <i>nn</i>	<i>nn</i> specifies the physical page size (number of lines) for TTY devices.
TTYPS=24	This is the default value.

UMODE - Session Processing Mode

This parameter determines the Natural session processing mode. Possible values are:

UMODE=(S,y)	S specifies the mode of operation. y is the language indicator. See explanation of operands below.
UMODE=(S,E)	This is the default value.

Explanation of Operands

Operand	Meaning:
S	Mode of operation = single; that is, one Natural session can be started from one terminal/user.
y	Language indicator for the restart message, the message for "free-running messages" and the logoff message. D=Danish E=English F=French G=German I=Italian N=Dutch S=Spanish

NATUTM Macro Entries

- CMKBADR - Current Address of UTM KB
- User Area in the Swap Pool Directory

CMKBADR - Current Address of UTM KB

The entry CMKBADR holds the current address of the UTM communication area (Kommunikationsbereich, KB).

The communication area can be accessed as shown in the following example, which illustrates an Assembler program that could be called from a Natural program.

Example:

```
EXAMPLE CSECT
STM 14,12,12(13)
USING EXAMPLE,15
L 2,VCONST LOAD ADDRESS OF KB-ADDRESS
L 3,0(,2) LOAD ADDRESS OF KB
.
.
LM 14,12,12(13)
BR 14
VCONST DC V(CMKBADR) ENTRY ADDRESS
END
```

In this case, the program name "EXAMPLE" must be defined with the keyword parameter LINK or LINK2 of macro NATUTM, and the program itself must be linked with the front-end part of the Natural UTM Interface.

User Area in the Swap Pool Directory

One fullword is available for user-defined purposes in the Natural swap pool directory - see label USERWRD in DSECT MEMPOOL of macro NAMSWDIR. This word can be used for synchronization, for example, for switching accounting on and off, whilst the Natural/UTM application is running.

The following example shows how this area can be addressed.

Example:

```
WXTRN CMKBADR ENTRY IN MACRO NATUTM
PROG CSECT
STM 14,12,12(13) SAVE REGISTERS
USING PROG,15 BASE OF PROGRAM
USING KB,4 BASE OF UTM KB
USING MAINDIR,5 BASE OF SWAP POOL DIRECTORY
L 3,KBADR LOAD ADDRESS OF KB ADDRESS
L 4,0(,3) LOAD ADDRESS UTM KB
L 5,ASWPDIR ADDRESS SWAP POOL DIRECTORY
OI USERWRD+3,1 SET THE LOW ORDER BIT OF FIELD
* USERWRD TO 1
LM 14,12,12(13) RELOAD REGISTERS
BR 14 RETURN
KBADR DC A(CMKBADR) ENTRY IN MACRO NATUTM
NAMSWDIR MACRO CALL FOR SWAP POOL DSECT
MAINDIR DSECT
```

```

.
.
USERWRD DS F DIRECTORY USER AREA
.
CMKBNEX MACRO CALL FOR UTM KB DSECT
KB DSECT
.
.
ADRSWAP DS F ADDRESS OF Natural SWAP POOL
.
END
    
```

When working in this area, the user must take care not to overwrite any other data in the swap pool directory. Mistakes could lead to abnormal termination of the UTM task.

NURENT Macro Keyword Parameters

The following parameters are available:

ACCNT | ATTKEY | AUTOLINK | CALLM31 | CLR3270 | EXTAPPL | FPUT | ILCS | K2 | PARMOD |
 SCRNRTRC | SPOOL | UINPEX | UOUTEX

ACCNT - Call Logic for User Account Routine

This parameter is used to define the logic for call of the user account routine (user exit ACCEXIT). Possible values are:

ACCNT=APPL	ACCEXIT is called at change of application (new Natural logon ID). This is the default value.
ACCNT=DIAL	ACCEXIT is called after every dialog step.

ATTKEY - Attention Interrupt Key

This parameter is used to define an attention interrupt key. Such a key definition only makes sense for output in non-conversational mode. Possible values are:

ATTKEY= <i>nnZ</i>	<i>nnZ</i> can be in the range of 26Z to 39Z.
ATTKEY=	Default value: no value

AUTOLINK - Use of AUTOLINK Function

This parameter specifies whether the AUTOLINK function of the dynamic binder/loader for loading of 3GL programs is activated or not. Possible values are:

AUTOLINK=YES	The AUTOLINK function is activated. This is the default setting.
AUTOLINK=NO	The AUTOLINK function is deactivated.

CALLM31 - Switching from 31 to 24-Bit Address Mode

This parameter is only relevant if Natural is generated for the 31-bit addressing mode and the front part is loaded below (PARMOD=31, see below).

CALLM31=YES	A call from a Natural program to a 3GL program will be executed in 31-bit addressing mode.
CALLM31=NO	Call in 24-bit mode. The addressing mode is switched from 31-bit to 24-bit before a 3GL program will be called from a Natural program. This is the default value.

Exceptions:

- The 3GL program is loaded above the 16-MB line.
- The address of the parameter list is above the 16-MB line.

CLR3270=xxx - Definition of CLEAR Key

This parameter defines the CLEAR key in the AID character table V (AID3270) for 3270-type devices (IBM).

CLR3270=xxx	xxx defines the CLEAR key.
CLR3270=PA1	By default, PA1 is the CLEAR key.

EXTAPPL - UTM TERMN Name of External DCAM or PDN Applications

This parameter defines the UTM TERMN name (see the parameter PTERM of UTM KDCDEF) of external DCAM or PDN applications. For these TERMN names, the MGET return code "05Z" (format changed) is ignored. Possible values are:

EXTAPPL=xx	xx and yy define the UTM TERMN name.
EXTAPPL=(xx,yy)	No default value is provided.

FPUT - Operation Supplement for Printing via KDCS-Call FPUT

This parameter defines the operation supplement for printing via KDCS-Call FPUT to a printer which is defined in the UTM KDCDEF.

FPUT=NE	Total print message.
FPUT=NT	Part of a print message.

ILCS - Support of CRTE or ILCS

This parameter specifies whether the common runtime environment for calls (CRTE) or the ILCS interface for calls of 3GL programs will be supported. Possible values are:

ILCS=CRTE	The common runtime environment for calls of 3GL programs will be supported. Prerequisite: The program "IT0SL#" must be included in the front part. INCLUDE IT0SL#,SYSLNK.CRTE.010 RESOLVE,SYSLNK.CRTE.010
ILCS=YES	Only the ILCS interface for calls of 3GL programs will be supported. Prerequisite: The program "IT0INITS" must be included in the front part. INCLUDE IT0INITS,SYSLNK.ILCS RESOLVE,SYSLNK.ILCS
ILCS=NO	CRTE or ILCS are not supported. This is the default value.

K2 - UTM Return Code for K2 Key

This parameter specifies the UTM return code for the K2 key (for Natural PA2)

K2= <i>nmn</i>	<i>nmn</i> must be in the range of 26Z to 39Z.
K2=27Z	This is the default value.

PARMOD - Generation of Front-End and Reentrant Parts

This parameter applies to the generation of both the front-end and reentrant parts. Possible values are:

PARMOD=(<i>nm,loc</i>) <i>nm</i> =24/31 <i>loc</i> =BELOW/ABOVE	The first part of this parameter (<i>nm</i>) is used to define an addressing mode (24-bit or 31-bit mode) for the Natural/UTM application. The second part of this parameter (<i>loc</i>) is used to define the front part location of the Natural/UTM application.
PARMOD=(31,ABOVE)	This is the default value.

PARMOD=(*nm,loc*) must also be defined for assembling macro NATUTM. Operands must be identical for the front-end and reentrant parts.

SCRNTRC - Tracing of Screen I/Os

This parameter is used for debugging screen I/O to find out the reason for certain error situations. If this parameter is set to ON/(ON,*nm*), a special debug buffer for each user will be allocated (default buffer size is 3 KB). Possible values are:

SCRNTRC=ON	A debug buffer for each user is allocated with a default buffer size of 3 KB.
SCRNTRC=(ON, <i>nm</i>)	A debug buffer for each user is allocated where <i>nm</i> is used to define a specific screen debug buffer size other than the default value of 3 KB.
SCRNTRC=OFF	This is the default value.

You should only set this parameter to ON/(ON,*nm*) after having consulted with Software AG Technical Support.

SPOOL - Automatic Start and Termination of Printer Task

This parameter enables you to specify a spooling system. The value for this parameter must be the same as the value for the SPOOL parameter in macro NATUTM. Possible values are:

SPOOL=(NATSPOOL,' <i>enter-parms</i> ', <i>n</i>)	For use with NATSPOOL (Natural Advanced Facilities), see Using NATSPOOL.
SPOOL=REPRO-2000	For use with a remote spooling system, see Using REPRO-2000 Remote Spooling System.
SPOOL=RMSPOOL	For use with your own user exit program, see Using RMSPOOL User Exit.

Default value: None.

UINPEX - Name of User Exit

With this parameter, you can specify the *name* of a user exit. This user exit is invoked by Natural/UTM after a terminal message has been sent; see also User Exits. Possible values are:

UINPEX= <i>name</i>	<i>name</i> specifies the name of a user exit
UINPEX=INPSCR	By default, user exit INPSCR is used.

UOUTEX - Name of User Exit

With this parameter, you can specify the *name* of a user exit. This user exit is invoked by Natural/UTM before a terminal message is to be sent; see also User Exits.

UOUTEX= <i>name</i>	<i>name</i> specifies the name of a user exit
UOUTEX=OUTSCR	By default, user exit OUTSCR is used.

Natural under UTM - Part 3

This part of the Natural UTM Interface documentation covers the following topics:

- User Exits
- Asynchronous Transaction Processing unter UTM
- Printing under UTM
- Calling Non-Natural Programs
- Calling UTM Chained Partial Programs
- Calling Adabas from Non-Natural Programs in a Natural UTM Application
- Terminating a UTM Task Abnormally

Installation - refer to Installing the Natural UTM Interface in the Natural Installation Guide for Mainframes.

Notation_vrs_or_vr">Notation *vrs* or *vr*: If used in the following document, the notation *vrs* or *vr* stands for the relevant *version*, *release*, *system maintenance level* numbers.

User Exits

Several user exits are provided in the Natural UTM Interface. These are described below.

To use any of these exits, the corresponding user program must be linked with the front-end part of the Natural/UTM application. The user exit RP2PRNT is an exception.

User exit routines are called with the customary register conventions.

ACCEXIT - Macro NATUTM

The user exit ACCEXIT can be used to retrieve accounting information. Depending on the value of the keyword parameter ACCNT in macro NURENT, this user exit is activated either at the end of each dialogue step or at each change of application (new Natural logon ID); see also Accounting for Natural UTM Applications .

ACCINIT - Macro NATUTM

The user exit ACCINIT can be used to gather accounting information. It is activated at the beginning of each dialogue step; see also Accounting for Natural UTM Applications .

INPTEX - Program FREXIT

The user exit INPTEX is activated whenever an input message is read. See also the description of the program INPTEX in the section Utility Programs for Use with Natural/UTM.

RP2PRNT - Macro NURENT

The user exit RP2PRNT is intended as an interface to other manufacturers' spooling systems. The user exit routine (spooling program) must be reentrant and linked with the reentrant part of the Natural/UTM application. See also Other Spooling Systems and the description of the keyword parameter SPOOL in the section Keyword Parameters of Macro NATUTM.

RMSPOOL - Macros NATUTM and NURENT

If you wish to write your own spooling interface program, call it "RMSPOOL". The program "RMSPOOL" can be linked to the (non-reentrant) front-end part or to the reentrant part of the Natural/UTM application. If it is to be linked to the reentrant part, the program itself must be written so as to be reentrant.

Important:

If program "RMSPOOL" is to be used, the SPOOL parameter in macros NATUTM and NURENT must be set to SPOOL=RMSPOOL.

The Natural UTM Interface passes the following parameters to program "RMSPOOL":

Address (Format/Length)	Contents
1st Address (A2)	Function code. Possible function codes are: OP - The print file has to be opened, and the first print record is passed. PR - Any subsequent print record is passed. CL - The print file has to be closed.
2nd Address	Print record (data to be printed). The first byte of the print record contains the line/form feed character. (If function code "CL", this is a dummy address.)
3rd Address (B2)	Length of print record (including feed character). (If function code "CL", this is a dummy address.)
4th Address (A8)	Printer name.
5th Address	Print buffer. This buffer can be used as work area by RMSPOOL (also if RMSPOOL is reentrant) for any purpose. The buffer is available for exclusive use by RMSPOOL between dialogue input and dialogue output.
6th Address (B2)	Length of print buffer.
7th Address (A8)	Current user ID (as in the system variable *USER).
8th Address (A8)	Current terminal ID (as in the system variable *INIT-ID).
9th Address (A8)	Current Natural library name (as in the system variable *LIBRARY-ID).
10th Address (A8)	Current Natural program name (as in the system variable *PROGRAM).
11th Address (A4/B4)	Return code. When RMSPOOL is invoked, the Natural UTM Interface sets this field to binary "0". Upon return of control from RMSPOOL, any value other than binary "0" is interpreted as error code and (if displayable) is displayed to the user on the terminal screen and also output to "SYSLST".

SHUTALL - Macro NATUTM

The user exit specified with the SHUTALL parameter in macro NATUTM is activated whenever a UTM task is terminated (KDCSHUT *n*). By default, this user exit is "SHUTEX1".

If the user exit specified with SHUTALL is to be used, the parameter USAGE=SHUT in KDCDEF for the Natural UTM Interface must have been set when generating KDCROOT.

SHUTLST - Macro NATUTM

The user exit specified with the SHUTLST parameter in macro NATUTM is activated when the *last* UTM task is terminated (KDCSHUT *n*). By default, this user exit is "SHUTEX2".

If the user exit specified with SHUTLST is to be used, the parameter USAGE=SHUT in KDCDEF for the Natural UTM Interface must have been set when generating KDCROOT.

STRTALL - Macro NATUTM

The user exit specified with the STRTALL parameter in macro NATUTM is activated whenever a UTM task is started. By default, this user exit is "STARTEX".

STRTFST - Macro NATUTM

The user exit specified with the STRTFST parameter in macro NATUTM is activated when the *first* UTM task is started. By default, this user exit is "STAPPLX".

TRMIOEX - Program FREXIT

The user exit TRMIOEX is activated with each formatted input or output message.

UINPEX - Macro NURENT

The user exit specified with the UINPEX parameter in macro NURENT is activated *after* a terminal message has been sent. By default, this user exit is "INPSCR".

Natural/UTM passes the following parameters to the user exit:

Address (Format/Length)	Contents
1st Address	Address input buffer.
2nd Address (B2)	Address message length.

UOUTEX - Macro NURENT

The user exit specified with the UOUTEX parameter in macro NURENT is activated *before* a terminal message is to be sent. By default, this user exit is "OUTSCR".

Natural/UTM passes the following parameters to the user exit:

Address (Format/Length)	Contents
1st Address	Address output buffer.
2nd Address (B2)	Address message length.

UVGEXIT - Macro NATUTM

The user exit UVGEXIT is activated at the start, restart and end (normal or abnormal) of a UTM DC transaction. The current task ID (Vorgangskennzeichen, KCKNZVG) is passed to the user exit routine.

WHCEXT - Macro NURENT

The user exit WHCEXT can be used to modify an output which is to be printed before it is passed by FPUT to UTM. When WHCEXT is called, register 9 contains the address of the output to be printed and register 13 the address of the save area.

WHCEXT must be reentrant and it must be linked to the reentrant part of the Natural/UTM application. For further information, please refer to the source listing of the assembled macro NURENT (Label 'NUWHC').

Asynchronous Transaction Processing under UTM

To start an asynchronous transaction, the service routine NATASYN in the Natural UTM Interface has to be called. The start of an asynchronous transaction in a Natural program is done by passing dynamic parameters according to the following pattern:

```

...
COMPRESS dynamic parameters INTO field
CALL 'NATASYN'[parameter area]
SET CONTROL 'H'
WRITE NOTITLE NOHDR field
[WRITE ...]
INPUT 'text' ifield (A1)
END

```

If the length of the dynamic parameters exceeds 250 bytes (that is, if more than one WRITE statement is required), a *parameter area* has to be passed with the CALL 'NATASYN' statement.

The parameter area is also required if the asynchronous transaction is to be started with UTM DPUT; that is, at a specific time. The aggregate length of the dynamic parameters must not exceed 3750 bytes. The *parameter area* for the CALL 'NATASYN' has the following structure:

Bytes	Contents
01-02	Number of WRITE statements.
03	DPUT time indicator: R = a relative time, A = an absolute time, <i>blank</i> = FPUT.
04-06	Day of the year.
07-08	Hours.
09-10	Minutes.
11-12	Seconds.

For the contents of bytes 03 - 12, the same rules apply as described for DPUT calls in the respective Siemens UTM documentation. Natural programming examples can be found in the Natural application SYSEXTP (programs STARTAS1, ASYNMULT, STARTAS, READAUTO, AWINDOW1, AWINDOW2).

For asynchronous transaction processing, KDCROOT, KDCDEF and the UTM startup job must be modified as necessary (see the Siemens UTM documentation).

All UTM TACs for asynchronous transactions must begin with the character sequence which is defined as a unique identifier for asynchronous TACs in parameter ASYNTAC of macro NATUTM. Conversely, the first five characters of UTM TACs for synchronous transactions must *not* be this character string.

Mixed transaction processing (that is, both within a single UTM application and between two UTM applications) is not possible.

Asynchronous Processing within a Natural UTM Application

If transactions are to be processed asynchronously within a Natural/UTM application, the operands of the keyword parameters SYAPPLI and ASAPPLI of macro NATUTM must be set to "NO" (this is the default value).

Example:

This is an example of a Natural program that initializes an asynchronous transaction within a Natural/UTM application.

```
* STARTAS - EXAMPLE OF THE INITIALIZATION FOR ASYNCHRONOUS
* TRANSACTION WORKING WITHIN ONE UTM APPLICATION
* PARMS ARE SEPARATED BY ', '
* SUBLIST IN STACK IS SEPARATED BY ';'
FORMAT LS=145
RESET PARM1(A144) PRDEST(A8) LTDEST(A8)
MOVE 'PRINTER1' TO PRDEST /* --> 1
MOVE *INIT-ID TO LTDEST /* --> 2
COMPRESS 'SENDER=' PRDEST ',OUTDEST=' LTDEST ', '
'MENU=F,STACK=(LOGON APPL1;READAUTO)'
INTO PARM1 LEAVING NO /* --> 3
CALL 'NATASYN' /* --> 4
SET CONTROL 'H' /* --> 5
WRITE NOTITLE NOHDR PARM1 /* --> 6
INPUT 'ASYNTASK INVOKED - HOPEFULLY' IFELD(A1) /* --> 7
END
```

Note	
1	The name (dummy) of a printer is moved into field PRDEST.
2	The logical name of the UTM terminal is moved into field LTDEST.
3	The message that is to be sent and processed by Natural is assembled, with the following information: the printer name (in this example, an arbitrary 8-character name), the logical name (KCLOGTER) of the terminal to which the message is to be sent, suppression of the main menu (this must be specified), the application name (Natural logon ID), the name of the program to be started and to be run in an asynchronous UTM transaction (READAUTO in the example).
4	When the subroutine NATASYN (in macro NATUTM) is called, a marker is set to indicate that an asynchronous transaction is to be initialized. The subroutine NATASYN conforms to the conventions for calling non-Natural programs.
5	The Natural offline report is activated.
6	The message (PARM1) is output by FPUT as an asynchronous transaction.
7	The Natural offline report is "switched off" by means of an INPUT statement that must have at least one input field.

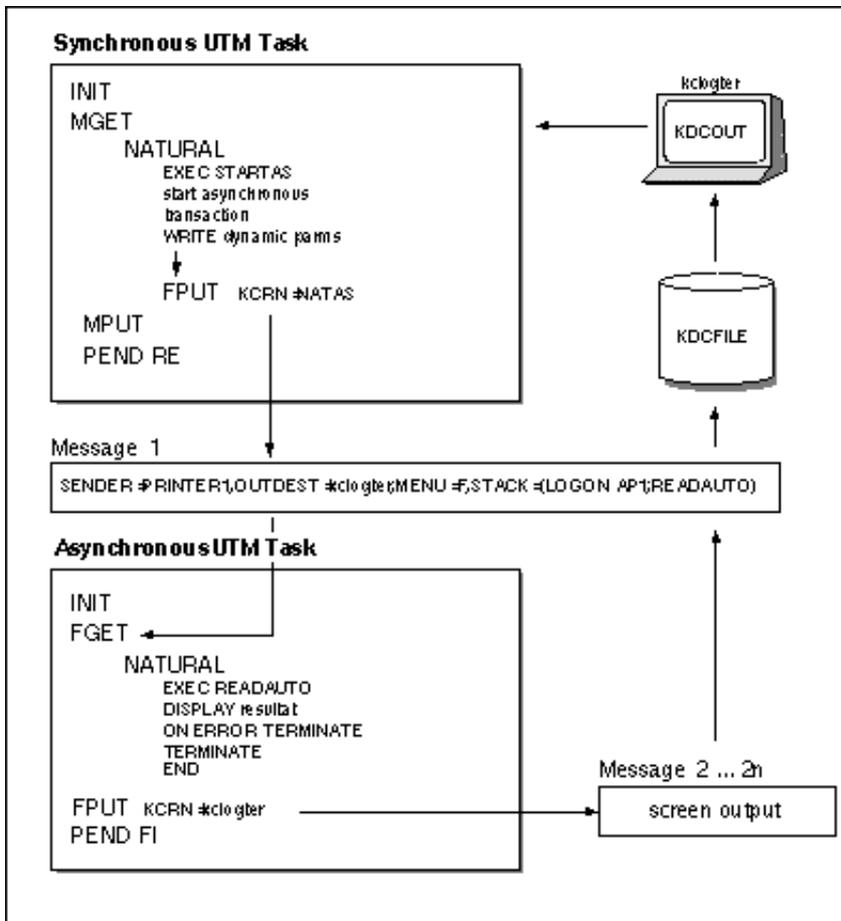
An example of the program that is to be executed asynchronously:

```

* READAUTO - EXAMPLE FOR ASYNCHRONOUS TRANSACTION WORKING
READ (75) AUTOMOBILES BY MAKE
WRITE MAKE MODEL HORSEPOWER YEAR
LOOP
ON ERROR DO /* --> 1
ERRNO(A4) = *ERROR
WRITE '*****'
/'ERROR NO.: ' ERRNO ' IN ASYNCHRONOUS PROGRAM ' *PROGRAM
/'*****'
TERMINATE
DOEND
TERMINATE /* --> 2
END
    
```

Note	
1	An "ON ERROR" routine must be defined in each program that is to be executed asynchronously. The routine must end with a TERMINATE statement.
2	Each program that is to be executed asynchronously must end with a TERMINATE statement.

Logic of an Asynchronous Transaction within one Natural UTM Application:



Asynchronous Processing between two Natural UTM Applications

When processing transactions asynchronously between two Natural/UTM applications, the logical UTM terminal name (LTERM name) of the synchronous application must be defined with the keyword parameter SYAPPLI of macro NATUTM, and the logical UTM terminal name (LTERM name) of the asynchronous application must be defined with the keyword parameter ASAPPLI of macro NATUTM.

Example:

```

NUSTART NATUTM SYAPPLI=LNATUTM,ASAPPLI=LNATASY,...
ASYNDRV NATUTM SYAPPLI=LNATUTM,ASAPPLI=LNATASY,...
    
```

KDCROOT and KDCDEF must be generated as appropriate for both applications.

Example for Synchronous Application:

```

OPTION GEN=ALL,ROOTSRC=INPUT.KDCROOT.KDCNATS
ROOT.KDCNATS
MAX KB=400,SPAB=8192,NB=5120,TRMSGLTH=520
MAX APPLINAME=NATUTM,APPLIMODE=S,KDCFILE=(NATUTM,S)
MAX TASKS=10,ASYNTASKS=5
.
.
EXIT PROGRAM=NUSTART,USAGE=START
EXIT PROGRAM=NUSTART,USAGE=SHUT
EXIT PROGRAM=FREXIT,USAGE=FORMAT
.
.
DEFAULT PROGRAM COMP=ASSEMB
PROGRAM NUSTART
PROGRAM FREXIT
PROGRAM NUERROR
.
.
DEFAULT TAC TYPE=D,PROGRAM=NUSTART,EXIT=NUERROR,CALL=BOTH,...
TAC NAT,ADMIN=NO,TIME=0
TAC NAT1,ADMIN=NO,TIME=0
.
.
DEFAULT TAC TYPE=A,PROGRAM=NUSTART,EXIT=NUERROR,CALL=FIRST,...
TAC NATSY
TAC NATAS
.
.
PTERM NATASY,PRONAM=HOST,PTYPE=APPLI,TERMN=A1,LTERM=LNATASI
DEFAULT PTERM PRONAM=PCDF,PTYPE=T9750,TRMN=FE,CONNECT=N,STATUS=ON
PTERM DFDSS001,LTERM=DF97501
PTERM DFDSS002,LTERM=DF97502
.
.
LTERM LNATASY
DEFAULT LTERM USAGE=D,STATUS=ON,ANNOAMSG=Y,RESTART=YES
LTERM DF97501
LTERM DF97502
.
.
SFUNC F1,RET=21Z
.
.
END

```

Example of Asynchronous Application:

```

OPTION GEN=ALL,ROOTSRC=INPUT.KDCROOT.KDCNATA
ROOT.KDCNATA
MAX KB=400,SPAB=8192,NB=5120,TRMSGLTH=520
MAX APPLINAME=NATASY,APPLIMODE=S,KDCFILE=(NATASY,S)
MAX TASKS=10,ASYNTASKS=5
.
.
EXIT PROGRAM=ASYNDRV,USAGE=START
EXIT PROGRAM=ASYNDRV,USAGE=SHUT
EXIT PROGRAM=FREXIT,USAGE=FORMAT
.
.
DEFAULT PROGRAM COMP=ASSEMB
PROGRAM ASYNDRV
PROGRAM FREXIT
PROGRAM NUERROR
.
.
DEFAULT TAC TYPE=D,PROGRAM=ASYNDRV,EXIT=NUERROR,CALL=BOTH,...
TAC NAT,ADMIN=NO,TIME=0
TAC NAT1,ADMIN=NO,TIME=0
.
.

```

Please see also the Siemens UTM documentations. If the asynchronous application is primarily intended for processing asynchronous transactions, storage can be saved by generating this application with a small (local) Natural swap pool of about 64 KB.

Important:

The TAC that was defined with the keyword parameter SYNTAC (the default value is "NATSY") must always be defined for KDCDEF with "TYPE=A"; this is an exception to the rules for naming UTM TACs. If, in addition, the synchronous application uses the UTM TACCLASS concept, an asynchronous TAC class must also be allocated to this TAC.

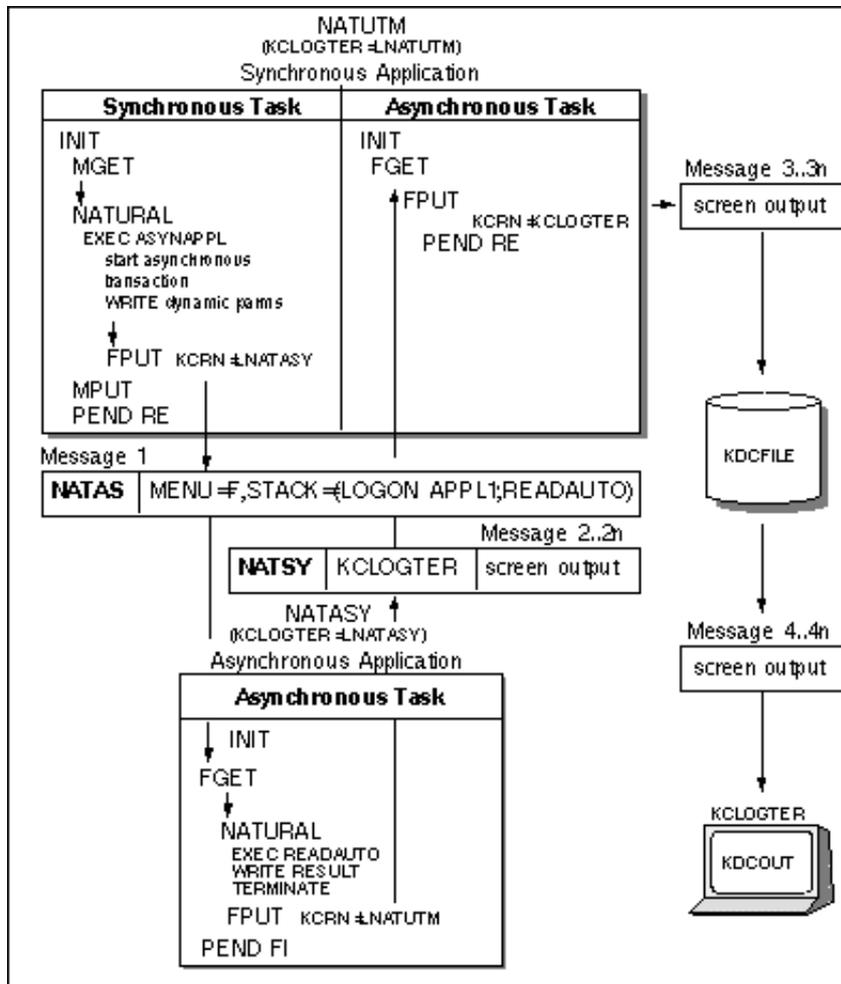
Example of a Natural Program to Initialize an Asynchronous Transaction between two Natural UTM Applications:

```
* ASYNAPPL - EXAMPLE OF INITIALIZATION FOR ASYNCHRONOUS
* TRANSACTION WORKING BETWEEN TWO UTM APPLICATIONS
FORMAT LS=145
RESET PARM1(A144) PRDEST(A8) LTDEST(A8) ASYNTAC(A8)
MOVE 'PRINTER1' TO PRDEST /* --> Note 1
MOVE *INIT-ID TO LTDEST /* --> Note 2
MOVE 'NATSY' TO ASYNTAC /* --> Note 3
COMPRESS 'NATAS' 'SENDER=' PRDEST ',OUTDEST=' LTDEST
',ASYNNAME=' ASYNTAC ',
'MENU=F,STACK=(LOGON APPL1;READAUTO)'
INTO PARM1 LEAVING NO /* --> Note 4
CALL 'NATASYN' /* --> Note 5
SET CONTROL 'H' /* --> Note 6
WRITE NOTITLE NOHDR PARM1 /* --> Note 7
INPUT 'ASYNTASK INVOKED - HOPEFULLY' IFELD(A1) /* --> Note 8
END
```

Note	
1	The name of a printer (simulation) is moved into the field PRDEST.
2	The logical name of the UTM terminal (KCLOGTER) is moved into the field LTDEST.
3	The standard TAC for sending "free-running" messages from the asynchronous application to the synchronous application is put in the field ASYNTAC. See also the description of the keyword parameter SYNTAC of macro NATUTM.
4	The message that is to be sent and processed by Natural is assembled, with the following information: transaction code for the asynchronous transaction (NATAS), printer name (in this example, an arbitrary 8-character string), the logical name (KCLOGTER) of the terminal to which the message is to be sent, the name of the standard TAC for sending free-running messages from the asynchronous application to the synchronous application, main menu suppression (this must be specified), the name of the application (Natural logon ID), the name of the Natural program to be started and to be run in the asynchronous transaction/application (in this example, READAUTO).
5	When subroutine NATASYN (in macro NATUTM) is called, a marker for the initialization of an asynchronous transaction is set. The subroutine conforms to the conventions for calling non-Natural programs.
6	The Natural offline report is activated.
7	The message (PARM1) is output with FPUT as an asynchronous transaction.
8	The Natural offline report is "switched off" by means of an INPUT statement with at least one input field.

The program to be executed asynchronously (READAUTO) must conform to the conventions that apply to asynchronous transaction processing within one Natural/UTM application.

Logic of Asynchronous Transaction between two Natural UTM Applications:



Printing under UTM

Using Local Non-Spooled Printers

A Natural program that wishes to use local printers without spooling (that is, with FPUT via UTM), should proceed as shown in the following example.

Example:

```
* TESTPRNT - TEST FOR THE Natural OFFLINE REPORT
RESET PARAM(A9)
REDEFINE PARAM (PARAM1(A1) PARAM2(A8))
MOVE 'H' TO PARAM1 /* --> Note 1
MOVE 'PRINTER1' TO PARAM2 /* --> Note 2
SET CONTROL PARAM /* --> Note 3
READ (50) AUTOMOBILES BY MAKE
WRITE NOTITLE NOHDR MAKE MODEL HORSEPOWER YEAR /* --> Note 4
LOOP
EJECT
INPUT 'PRINT ORDER WAS EXECUTED' IFELD(A1) /* --> Note 5
END
```

Note	
1	The Natural offline report is activated by putting an "H" in the first position of the field PARAM.
2	The logical UTM printer name is defined starting at the second position of the field PARAM.
3	The SET CONTROL statement, together with the content of the field PARAM, activates the Natural offline report and specifies the name of the printer. To ensure compatibility for existing programs written using Natural Version 1, the programs CMLIST and NATPRNT continue to be available; see Utility Programs for Use with Natural/UTM.
4	The print records are passed to UTM from the Natural UTM Interface using FPUT.
5	The INPUT statement (which must have at least one input field) deactivates the Natural offline report.

All the necessary steps for using local printers must have been taken when generating UTM; for further details, please refer to the Siemens UTM documentation. The appropriate UTM administration commands can be used to verify that a connection to the defined printers exists.

Using NATSPOOL (Natural Advanced Facilities)

The keyword parameter SPOOL of macro NATUTM is provided for using NATSPOOL under UTM. Further details are given in the section Keyword Parameters of Macro NATUTM. Please refer also to the BS2000/OSD-specific installation information in the Natural Advanced Facilities documentation.

If in an asynchronous UTM transaction printing is to be done with NATSPOOL, the TERMINATE statement must be preceded by an END OF TRANSACTION statement.

Other Spooling Systems

The user exit RP2PRNT is provided for interfacing to other spooling systems. This user exit is activated if "REPRO-2000" is specified with the keyword parameter SPOOL in macro NURENT. (This value should be used for all spooling systems.)

Since it must be linked with the reentrant part of the Natural/UTM application, the user exit routine RP2PRNT must be reentrant.

The logic of the transfer of print records from Natural, buffer processing, etc., can be seen in macro NURENT (labels CMWTERM and CMWHC) and the appropriate routines in macro NATUTM.

As an alternative, it is possible to use the user exit RMSPOOL; see User Exits.

Software AG does not provide support for this interface to other spooling systems except as described in the preceding paragraphs.

Calling Non-Natural Programs

Non-Natural programs are called using the standard register conventions for inter-program communication. If the program to be called is reentrant (uses shared code), it can be defined with `CSTATIC` in the Natural parameter module (macro `NTPRM`) and linked with the reentrant part of the UTM application. Otherwise, one of the following procedures can be used:

- The programs can be dynamically loaded at runtime.
To do this, the programs must be in the library defined by the `LIBNAM` parameter in the Natural parameter module or in the `BLSLIB` libraries specified in the UTM start job;
- The programs can be linked with the front-end part of the Natural/UTM application. To do this, the names of the programs must be defined in the operand of the keyword parameter `LINK`, `LINK2`, `LINK3` or `LINK4` of macro `NATUTM`.
This procedure is always necessary for programs that contain an `EXTRN` reference to an `ENTRY` that is already present in the front-end part of the Natural UTM Interface. The Natural UTM Interface executes a `TABLE` macro call for the programs that have been defined in this way. This makes an entry in the dynamic loader's `LINK` table, indicating that it is not necessary to dynamically load the programs when they are called by the Natural program.

In both cases, the maximum number of called non-Natural programs must be defined with the keyword parameter `CDYNAM` of macro `NATUTM`; see `Keyword Parameters of Macro NATUTM`.

Attention: If keyword parameter `KB` in macro `NATUTM` is set to "YES", Natural always passes the address of the UTM communication area (Kommunikationsbereich, `KB`) as the first parameter address. This does not apply to programs which are defined with `CSTATIC`.

Calling UTM Chained Partial Programs

Several methods are provided for ending a Natural session (FIN or TERMINATE) with a PEND PR(OGRAM) instead of a PEND FI(NISH), so that another UTM partial program is called:

- The UTM TAC for the UTM partial program that is to be called can be passed using the Natural dynamic parameter PROGRAM at the start of the Natural session.
Example: STACK= (LOGON APPL1 ; MENU) , PROGRAM=NAT10
- The UTM TAC for the UTM partial program that is to be called can be defined in the operand of the keyword parameter PENDPR of macro NATUTM.
Example: NATUTM PENDPR= ' NAT10 '
- The utility program TACSWTCH can be used.

In all cases, the Natural UTM Interface would execute a PEND PR(OGRAM) with the UTM TAC NAT10 at the end of the Natural session, which means that the UTM partial program associated with this TAC would be started.

Another way to execute a PEND PR(OGRAM) is by activating the function key defined for this purpose, which suspends, but not terminates, the Natural session. On return from the UTM partial program with PEND PR(OGRAM), the Natural session can be continued from the point at which it has been suspended; see also the keyword parameter PRKEY. If the function key for PEND PR(OGRAM) is activated without a UTM TAC for another UTM partial program being defined, an appropriate error message is displayed.

Note:

The programs NUEXAMPL, UTMNAV and UTMCOB show examples of the logic necessary in a UTM partial program that wishes to communicate with the Natural UTM Interface (and therefore with Natural itself) - see the descriptions of programs UTMCOB and UTMNAV in the section Software Exchange.

Calling Adabas from Non-Natural Programs in a Natural UTM Application

If a Natural program calls a non-Natural program that also includes Adabas calls, the appropriate field in the Adabas control block must be supplied with the current Adabas user ID. In this case, generate the CSECT ADACALL in the Natural UTM Interface.

ADACALL contains an entry which is defined with the keyword parameter ADACALL in macro NATUTM (the default value of this parameter is "Adabas").

This entry is activated for every "CALL [Adabas] USING ...". The current Adabas user ID is passed to the field ADDITIONS2 of the Adabas command block, and subsequent processing of the Adabas call is passed to the Adabas interface module ADALNN; see also the keyword parameter ADACALL.

Terminating a UTM Task Abnormally

The Natural session (and thereby also the UTM task) can be abnormally terminated by entering the CANCEL parameter's value of the Natural parameter module (default is *CANCEL in upper-case letters).

Natural under UTM - Part 4

This part of the Natural UTM Interface documentation covers the following topics:

- Accounting for Natural UTM Applications
- Utility Programs for Use with Natural/UTM
- Software Exchange
- UTM TACCLASS Concept - Priority Control
- Generating a Natural UTM Application
- Optimizing Natural UTM Applications
- Several Applications with one Common Natural
- Entering and Defining Dynamic Natural Parameters
- UTM User Restart
- Adabas Priority Control

Installation - refer to Installing the Natural UTM Interface in the Natural Installation Guide for Mainframes.

Notation *vrs* or *vr* -> **Notation *vrs* or *vr***: If used in the following document, the notation *vrs* or *vr* stands for the relevant *version*, *release*, *system maintenance level* numbers.

Accounting for Natural/UTM Applications

To better control the use of resources by Natural/UTM applications, accounting records are made available by the user exits ACCINIT and ACCEXIT.

The user exit ACCINIT is activated by the Natural UTM Interface at the beginning of each dialogue step.

The user exit ACCEXIT is activated by the Natural UTM Interface depending on the keyword parameter ACCNT in macro NURENT:

ACCNT=DIAL	The user exit ACCEXIT is activated at the end of each dialogue step.
ACCNT=APPL	The user exit ACCEXIT is activated at each change of application (new Natural logon ID).

In both cases, an accounting record is also provided at the end of the session (FIN system command or TERMINATE statement).

Structure of the Accounting Record

0 - 7	Logical UTM terminal name	DS	CL8
8 - 15	User ID	DS	CL8
16 - 23	Current Natural application name	DS	CL8
24 - 27	Number of Adabas calls	DS	F
28 - 31	Accumulated message length	DS	F
32 - 35	Elapsed time in Natural including subroutines (milliseconds)	DS	F
36 - 37	Number of pages printed	DS	H
38 - 39	Number of terminal I/O transfers	DS	H
40 - 49	(user area)	DS	CL10
50 - 51	unused	DS	CL2
52 - 55	Adabas command time (milliseconds)	DS	F
56 - 63	Name of last transaction program	DS	CL8

The user area of the accounting record can (if required) be used for additional application-specific accounting information. The accounting area is in the user-specific UTM communication area (Kommunikationsbereich, KB).

The current address of the UTM KBs can be found with the entry "CMKBADR" of macro NATUTM as necessary; otherwise, the operand of the keyword parameter KB of macro NATUTM must be set to "YES". In this case, Natural passes the address of the communication area as the first parameter of every subroutine call.

The user exit routine ACCEXIT can store the accounting records in an Adabas file, in a shared sequential PAM dataset or in a task-specific SAM dataset. The program ACCEXIT shows an example of the method for storing accounting records; see Software Exchange.

Utility Programs for Use with Natural/UTM

Several utility programs are provided for use with Natural under UTM. The following rules apply to their usage:

- The Natural and UTM macro libraries must be used when assembling these utilities.
- When a particular program is to be used:
 - its name must be specified with the keyword parameter LINK or LINK2 of macro NATUTM
 - and the program itself must be linked with the front-end part of the Natural/UTM application.

A detailed description, including the interface, valid parameter values and a summary of the logic, can be found in each program's maintenance log.

Utility Program NATDUE

The program NATDUE can be used to find out within a Natural program whether the user has entered data in the current dialogue step or whether merely EM/DÜ or DÜ was entered.

The utility program INPTEX must be used if NATDUE is to be called. The program INPTEX satisfies the user exit INPTEX in the format exit module FREXIT and checks at each dialogue step whether data were entered. According to the result of this test, a flag that is subsequently interrogated by the program NATDUE is set in the communication area (Kommunikationsbereich, KB).

Example of a Natural Program that Calls NATDUE:

```
* PROG1 - EXAMPLE FOR CALLING THE SUBROUTINE 'NATDUE'
RESET P1(A1) ...
...
INPUT USING MAP ...
CALL 'NATDUE' P1
IF P1 = 'Y' DO ... /* INPUT FROM USER
IF P1 = 'N' DO ... /* NO INPUT FROM USER
IF P1 = 'E' DO ... /* ERROR
...
END
```

Utility Program INPTEX

The utility program INPTEX satisfies the user exit of the same name in the format exit FREXIT.

Important: INPTEX must be linked with the front-end part of the Natural/UTM application.

Warning: Any modifications that can be made to this program, for example, ignoring data entered in a particular line on the terminal screen, are made at the user's risk.

Function: This program checks each input message for the presence of input from the terminal, or whether merely EM/DÜ or DÜ was pressed.

It is not necessary to define the program name INPTEX with the keyword parameter LINK or LINK2 of macro NATUTM.

Utility Program NATPRNT

The program NATPRNT provides the following special service functions for operating local printers:

- accepting the logical name of the target printer;
- verifying the printer name against a list of valid printer names;
- setting a marker for building variable length print records.

Utility Program UTMTAC

The program UTMTAC, which can be called from a Natural program, yields the current UTM TAC. This makes it possible for a central Natural program to perform UTM TAC-controlled "navigation" within a Natural/UTM application.

Utility Program TACSWTCH

The utility program TACSWTCH is a macro which can be used to dynamically assign a UTM TAC for a PEND PR(OGRAM) from within a Natural program. The specified UTM TAC is checked against the generated UTM table and saved accordingly. Also, information can be passed to the PEND PR(OGRAM). To use this utility, proceed as follows:

1. Define the valid UTM TACs and assemble the TACSWTCH macro:

For Example: TACSWTCH TAC=(tac1,tac2,tac3,...tacn)

These TACs have to be defined in KDCDEF as well, and for the generation of KDCROOT they have to be assigned to the corresponding UTM partial programs.

1. Define the program TACSWTCH with the keyword parameters LINK to LINK4 in macro NATUTM.
2. Link program TACSWTCH to the front-end part of the Natural UTM Interface.
3. Interface description: CALL 'TACSWTCH' P1 [P2] P3

P1 (A8)	Contains the UTM TAC to be used for a PEND PR.
P2 (An)	Is optional and contains the length and data of a message to the PEND PR The structure of P2 is: LLLDDD..... LLL = Message length (3 digits, no length field); minimum length: 000, maximum length: 160. DDD = Message area.
P3 (A1)	Has two functions: On call and if P3 contains the value "G" (Go), the PEND PR(OGRAM) is executed at the next Natural output (INPUT, WRITE, DISPLAY). After calling the Natural UTM Interface with PEND PR, the Natural session is continued where it had been suspended, which means that the last output is displayed to the user. On return, P3 contains the return code from TACSWTCH. Possible return codes are: 0 = The operation has been executed without error. 1 = TAC has not been found in the TAC table. 2 = Message length was less than "000". 3 = Message length was over "160". Once TACSWTCH has been called without error, a PEND PR(OGRAM) can be executed by either issuing a FIN command or with a TERMINATE statement or by activating the function key for PEND PR; see the keyword parameter PRKEY.

Special TACSWTCH Functions

You can use the first TACSWTCH parameter with the following values:

RESET	The UTM TAC currently available will be cleared, that is, the session will be terminated with PEND FI.
GETP	Data will be moved from the print buffer to the adequate data area of the calling Natural program.
GETU	Data will be moved from the KB user extension to the adequate data area of the calling Natural program.

The first two bytes (format: binary) in the print buffer or in the KB user extension must contain the data length (including these first two bytes).

PUTP	Data will be moved from the adequate data area of the calling Natural program to the print buffer.
PUTU	Data will be moved from the adequate data area of the calling Natural program to the KB user extension.

The first two bytes (format: binary) in the data area of the Natural program must contain the data length (including these first two bytes). The data will be moved including the first two bytes.

Example for PUTP and GETP:

```

DEFINE DATA LOCAL
01 P1(A8) /* FUNCTION CODE/UTM TAC
01 P2(A252) /* FIRST PART OF DATA AREA
01 REDEFINE P2
02 P21(B2) /* DATA LENGTH INCLUDING FIRST TWO BYTES
02 P22(A250)
01 A1(A250) /* SECOND PART OF DATA AREA
01 P3(N1) /* RETURN CODE
END-DEFINE
... /* PROGRAM LOGIC
MOVE 'PUTP' TO P1 /* MOVE FUNCTION CODE FOR TACSWTCH
MOVE 502 TO P21 /* MOVE TOTAL LENGTH OF DATA
CALL 'TACSWTCH' P1 P2 P3 /* PUT DATA INTO PRINT BUFFER
IF P2 NE 0 /* RETURN CODE CONTROLLING
DO... /* ERROR LOGIC
MOVE 'NAT1' TO P1 /* MOVE ADEQUATE UTM TAC
MOVE 'G' TO P3 /* EXECUTE PEND PR WITH TAC NAT1
CALL 'TACSWTCH' P1 P3
IF P3 NE 0 /* RETURN CODE CONTROLLING
DO... /* ERROR LOGIC
INPUT ' ' /* DUMMY MESSAGE FOR DRIVER CONTROL

```

Now the Natural/UTM driver gets control and runs with the following logic:

1. It ignores the dummy message (INPUT ' ').
2. MPUT with LENGTH=0 and PEND PR with TAC 'NAT1' for the UTM partial program.
3. The UTM partial program gets the Natural program data through the print buffer. The print buffer is located in the UTM SPAB and the address of the print buffer is defined in the field 'KBAPBUFF', which is located in the UTM KB:
 - It moves data for the Natural program into the print buffer (the first two bytes must contain the data length in binary format, including the two-byte length field).
 - It executes an MPUT with LENGTH=0 and a PEND PR with the TAC defined for the Natural/UTM driver.
4. The Natural/UTM driver gets control (INIT/MGET).
5. It simulates ONLY ENTER for Natural.
6. It resumes with Natural as follows:

```

MOVE 'RESET' TO P1 /* MOVE FUNCTION CODE FOR TACSWTCH
CALL 'TACSWTCH' P1 P3 /* RESET PEND PR TAC (NAT1)
IF P3 NE 0 /* RETURN CODE CONTROLLING
DO... /* ERROR LOGIC
MOVE 'GETP' TO P1 /* MOVE FUNCTION CODE FOR TACSWTCH
CALL 'TACSWTCH' P1 P2 P3 /* GET DATA FROM PRINT BUFFER
IF P3 NE 0 /* RETURN CODE CONTROLLING
DO... /* ERROR LOGIC
... /* PROGRAM LOGIC
END

```

If the keyword parameter **KBSAVE** of macro **NATUTM** is set to **YES**, the called UTM partial program may use the UTM KB (from the end of the header plus first twelve bytes). In this case, the UTM KB will be saved (beginning from KB header plus first twelve bytes) with **SPUT** and will be refreshed with **SGET**.

When defining UTM transaction codes for the transaction logic between Natural and other UTM partial programs, the following rule applies:

For a **PEND PR(ogram)** from another UTM partial program to the Natural/UTM driver, the preceding start TAC may never be used. The fact that the Natural/UTM driver was called by a **PEND PR(ogram)** can only be recognized if the contents of the preceding start TAC in field **KCTACVG** are different from the current TAC in field **KCTACAL**. (Normally, field **KCTACVG** contains the TAC with which the user has entered the application.)

Software Exchange

Software AG's customers have developed programs that meet certain specific needs found in their Natural/UTM applications. These programs are made available to all interested users via the "Software Exchange". This also applies to programs developed by Software AG that demonstrate example solutions to particular problems.

These programs, which are available free of charge, are not maintained by Software AG. The documentation of each program is usually included in the maintenance log of the source listing.

Program XAMDUSA

This program saves and restores the current user-specific WORKING-STORAGE SECTION of the calling COBOL program.

This enables user-specific data areas, for example tables, to be accessible over many dialogue steps and without regard to the UTM task in which the user is currently running. The data are saved in a PAM file using logical/physical chained PAM-I/O.

Program UTMCOB

Program UTMCOB is an example of a user-specific UTM partial program within a Natural/UTM application. It shows the fundamental logical structure of a program that, as a UTM partial program:

- Can be activated by the user by associated UTM TACs.
- Activates the Natural UTM Interface and hence the Natural application by means of PEND PR(OGRAM) with dynamic Natural parameters.
- Can be activated from the Natural UTM Interface by means of PEND PR(OGRAM).

See also Calling UTM Chained Partial Programs.

Program UTMNAV

Program UTMNAV is another example of a user-specific UTM partial program within a Natural/UTM application:

- It can be activated by the user or with PEND PR(OGRAM) by the associated UTM TAC.
- It interprets passed messages as dynamic Natural parameters.
- It provides screen output of information on the program logic.
- Previously received screen input (Natural dynamic parameters) is sent with MPUT and passed to the Natural UTM Interface with PEND PR(OGRAM).

Program UTMNAV contains an example of how the UTM KB can be used as a "common" user area.

Program NUEXAMPL

Program NUEXAMPL is an example of a user-specific UTM partial program which can exchange data with a Natural program. The program logic of NUEXAMPL and of the calling Natural program is described in the maintenance log of NUEXAMPL.

Program ACCEXIT

Program ACCEXIT is an example of a program that saves accounting data on a shared ISAM dataset. The user exits ACCEXIT and SHUTEX2 of the Natural UTM Interface are used. See also Accounting for Natural/UTM Applications.

Program TABMOD

The program TABMOD, which can be called from a Natural program, performs the following functions:

- load data records, for example a table, into a common memory pool using a unique key when an application is started and whilst an application is running;
- transfer data records according to the requirements of the calling Natural program.

This makes it possible to load frequently-needed data into storage once only and then keep them resident.

TABMOD is available as a macro in the library NUT nnn .MAC. It contains all information necessary for its installation and usage.

UTM TACCLASS Concept - Priority Control

Natural programs can allocate UTM TAC classes to optimize resource control using the UTM TACCLASS concept in a Natural/UTM application.

The following procedure should be followed when generating the Natural/UTM application and creating the Natural program:

Step 1: Specify UTM TACs and TAC Classes in the KDCDEF and KDCROOT Definitions

```

Example:

OPTION GEN=ALL,ROOTSRC=INPUT.KDCROOT.KDCNATP
ROOT KDCNATP
MAX APPLINAME=NATUTM,APPLIMODE=S,KDCFILE=(NATUTM,S)
MAX KB=400,SPAB=8192,NB=5120,TRMSGLTH=5120
MAX TASKS=10
MAX ASYNTASKS=3
...
EXIT PROGRAM=NUSTART,USAGE=START
EXIT PROGRAM=NUSTART,USAGE=SHUT
EXIT PROGRAM=FREXIT,USAGE=FORMAT
...
DEFAULT PROGRAM COMP=ASSEMB
PROGRAM NUSTART
PROGRAM FREXIT
PROGRAM NUERROR
PROGRAM KDCADM,COMP=SPL4
...
DEFAULT TAC TYPE=D,PROGRAM=NUSTART,EXIT=NUERROR,CALL=BOTH,...
TAC NAT,TIME=(3600000,5400),TACCLASS=1,...
TAC NAT1,TIME=(3600000,5400),TACCLASS=2,...
...
DEFAULT TAC TYPE=A,PROGRAM=NUSTART,EXIT=NUERROR,CALL=FIRST,...
TAC NATAS,TACCLASS=9
TAC NATAS1,TACCLASS=10
...
TACCLASS 1,TASKS=3
TACCLASS 2,TASKS=1
TACCLASS 9,TASKS=2
TACCLASS 10,TASKS=1
...
END
    
```

See also the Siemens documentation "UTM Generierung und Administration" (UTM Generation and Administration).

Notes on the UTM TACs Defined

NAT	This is the UTM TAC for less resource-intensive synchronous transactions; that is, transactions of short duration.
NAT1	This is the UTM TAC for more resource-intensive synchronous transactions; that is, transactions of longer duration.
NATAS	This is the UTM TAC for less resource-intensive asynchronous transactions.
NATAS1	This is the UTM TAC for more resource-intensive asynchronous transactions.

Step 2: The Structure of the UTM Start Job

The name of the job is "EN.NATUTM".

Example:

```
/.NATUTM LOGON Natural,E,,TIME=10000
/SYSFILE SYSOUT=PROT.UTMSTAT
/FILE NATUTM.KDCA,LINK=KDCFILE
/ERASE NATUTM.PRINTCONTROL
/STEP
/FILE LOG.NATUTM,LINK=SYSLOG
/FILE NATUTM.SWAPFILE,LINK=PAMNAT,SHARUPD=Y
/SYSFILE TASKLIB=NAT210.MOD
/.REPEAT EXEC NATUTM.E
.UTM START FILEBASE=NATUTM
START TASKS=7
START ASYNTASKS=3
START STARTNAME=EN.NATUTM
.UTM END
/SKIP .REPEAT
/STEP
/SYSFILE SYSOUT=(PRIMARY)
/STEP
/SYSFILE SYSLST=(PRIMARY)
/CAT NATUTM.PRINTCONTROL,SHARE=YES
/PRINT LST.NATUTM.,SPACE=E
/ERASE LST.NATUTM.
/STEP
/LOGOFF NOSPOOL
```

Step 3: Change the TAC Class of Synchr. Transactions by a Natural Program

The TAC-class of synchronous UTM transactions can be changed by a Natural program with the statements:

```
CALL 'NATTAC' operand1 [operand2] [operand3]
INPUT 'TACCLASS'
```

<i>operand1</i>	Must contain the value "S= <i>n</i> ", where "S" denotes "synchronous" and " <i>n</i> " is an integer value (0 - 4) that denotes the priority level of the transaction in subroutine NATTAC's table of transaction codes for synchronous TACs.
	If " <i>n</i> " is 0, the table of transaction codes is not used. The TAC to be used is passed explicitly in <i>operand2</i> when NATTAC is called.
	If " <i>n</i> " is a value in the range 1 - 4, the priority level of the desired TAC is taken from the appropriate keyword parameter TCLS1 - TCLS4 (for synchronous transactions) or TCLA1 - TCLA4 (for asynchronous transactions).
	If the subroutine NATTAC detects an error in <i>operand1</i> , it returns immediately to the calling program with an error code in <i>operand1</i> :
	E01: The first two characters of <i>operand1</i> were neither "S=" nor "A=".
	E02: The third character of <i>operand1</i> was <0 or >4.
	E03: No UTM TAC was defined for the specified priority level when the Natural/UTM application was generated, which means that the corresponding keyword parameter (TCLSn or TCLA <i>n</i>) has the value "-".
<i>operand2</i>	Optional. Must contain the UTM TAC for the desired TAC class if the third character of <i>operand1</i> is "0".
<i>operand3</i>	Optional. Must contain the value "Y" if the current user's subsequent dialogue is to be executed with the UTM TAC defined in <i>operand1</i> or <i>operand2</i> . If <i>operand3</i> is omitted when NATTAC is called, or if <i>operand3</i> has some value other than "Y", the START transaction code for the current user is used again with the first terminal output (standard function). If <i>operand3</i> has the value "Y" when NATTAC is called, further processing for the current user takes place with the UTM TAC specified in <i>operand1</i> (implicit) or <i>operand2</i> (explicit).

The statement INPUT 'TACCLASS' does not perform any terminal I/O; its function is merely to control the TACCLASS allocation.

Alternatively, a Natural program can call the Natural subprogram "NATTAC" with a CALLNAT statement. For this, the INPUT 'NATTAC' statement is omitted; the operands are the same as for the CALL statement (see above):

```
CALLNAT 'NATTAC' operand1 [operand2] [operand3]
```

This procedure can be used with synchronous as well as asynchronous transactions. NATTAC is contained in the library SYSTEM.

Example 1:

A Natural program that allocates a UTM TAC explicitly to assign a new TAC class and then changes over to the START UTM TAC.

```

* TACCLASS - EXAMPLE FOR A TACCLASS SWITCH
RESET CONTROL(A3) NEWTAC(A8) NR(N3)
REDEFINE CONTROL (ERRFLD(A1))
INPUT 'TEST FOR A TACCLASS SWITCH - NEW TAC: NAT1' IFELD(A1)
MOVE 'S=0' TO CONTROL /* SYNCHR. TAC, EXPLICIT --> Note 1
MOVE 'NAT1' TO NEWTAC /* SET NEW TAC --> Note 2
CALL 'NATTAC' CONTROL NEWTAC /* INVOKE TAC SWITCH --> Note 3
IF ERRFLD = 'E' DO /* ERROR CHECK --> Note 4
DISPLAY 'ERROR' CONTROL 'FROM NATTAC'
TERMINATE
DOEND
INPUT 'TACCLASS' /* ACTIVATE NEW TAC --> Note 5
READ (50) AUTOMOBILES BY MAKE /* NOW IN NEW TACCLASS --> Note 6
ADD 1 TO NR
WRITE NOTITLE NOHDR NR MAKE MODEL /* START TAC IS USED --> Note 7
LOOP
ON ERROR DISPLAY 'ERROR IN PROGRAM TACCLASS'
END
    
```

Note	
1	The value "S=0" indicates that it is a synchronous transaction and that the TAC is passed explicitly in the second parameter of the CALL 'NATTAC', which means that the TAC table is not used.
2	The new TAC (NAT1) is set up for the call to NATTAC.
3	The change of TAC class is initialized by calling NATTAC.
4	An error check is performed after returning from subroutine NATTAC.
5	A "pseudo"-MPUT and a "PEND PA" are executed with the new TAC.
6	The program is now running in the TAC class for NAT1.
7	When the first terminal output starts, the START UTM TAC takes effect again.

In this example, the AUTOMOBILE file is read using the UTM TAC NAT1. When the first terminal output begins, the START UTM TAC (NAT) takes effect again.

Internal Processing Logic: When NATTAC is called, a flag is set in the UTM communication area (Kommunikationsbereich, KB) indicating that a change of TACCLASS is pending.

The UTM TAC passed by the program is also stored in the user-specific communication area. The operation INPUT 'TACCLASS' causes terminal output from Natural, which causes the UTM interface to issue an MPUT and a PEND 'PA' with the new UTM TAC (the message is received by the Natural UTM Interface itself). When the message is received (in the new TAC class), the presence of the TACCLASS change flag causes the interface to simulate an ETX/DÜ in its input area. Further processing runs in the new TAC class.

Depending upon the value of the operand in the previous call of NATTAC, the first message sent to the terminal can cause an MPUT and a PEND 'PR' with the user's START UTM TAC; that is, a further TACCLASS change may take place.

Example 2:

A Natural program that allocates a UTM TAC explicitly to assign a new TAC class without changing over to the START UTM TAC.

```
* TACCLAS1 - EXAMPLE FOR A TACCLASS SWITCH
RESET CONTROL(A3) NEWTAC(A8) SWOFF(A1)
INPUT 'TEST FOR A TACCLASS SWITCH - NEW TAC: NAT1' IFELD(A1)
MOVE 'S=0' TO CONTROL /* SYNCHR. TAC, EXPLICIT
MOVE 'NAT1' TO NEWTAC /* SET NEW TAC
MOVE 'Y' TO SWOFF /* NO RESET TO START TAC
CALL 'NATTAC' CONTROL NEWTAC SWOFF /* INVOKE TAC SWITCH
INPUT 'TACCLASS' /* ACTIVATE NEW TAC
FETCH 'TACCLAS2' /* NOW IN NEW TACCLASS
END
* TACCLAS2 - THIS PROGRAM IS FETCHED FROM PROGRAM TACCLAS1
RESET NR(N3)
READ (25) AUTOMOBILES BY MAKE /* TACCLASS IS NAT1
ADD 1 TO NR
WRITE NOTITLE NOHDR NR MAKE MODEL HORSEPOWER YEAR
LOOP
FETCH 'MAINMENU' /* TACCLASS = NAT1
END
```

In this example, processing is assigned to a new TAC class with TAC NAT1. Switching to the user's START UTM TAC is avoided by the presence of the third parameter (SWOFF) in the call to NATTAC with value "Y".

It is also possible to perform several TACCLASS changes within one Natural program.

Example 3:

A Natural program that performs two explicit and one implicit TACCLASS changes.

```
*TACMULT - EXAMPLE FOR TWO TACCLASS SWITCHES IN ONE PROGRAM
RESET CONTROL(A3) NEWTAC(A8) SWOFF(A1) NR(N4)
INPUT 'TEST FOR 2 TACCLASS SWITCHES' IFELD(A1)
MOVE 'S=0' TO CONTROL /* SYNCHR. TAC, EXPLICIT
MOVE 'NAT1' TO NEWTAC /* SET NEW TAC
MOVE 'Y' TO SWOFF /* NO RESET TO START TAC
CALL 'NATTAC' CONTROL NEWTAC SWOFF /* INVOKE TAC SWITCH
INPUT 'TACCLASS' /* ACTIVATE NEW TAC
READ (50) AUTOMOBILES BY MAKE /* NOW IN NEW TACCLASS
ADD 1 TO NR
WRITE NR MAKE MODEL YEAR
LOOP
EJECT /* ACTIVATE NEW OUTPUT *****
MOVE 'S=0' TO CONTROL /* SYNCHR. TAC, EXPLICIT
MOVE 'NAT2' TO NEWTAC /* SET NEW TAC
CALL 'NATTAC' CONTROL NEWTAC /* INVOKE TAC SWITCH
INPUT 'TACCLASS' /* ACTIVATE NEW TAC
READ (100) AUTOMOBILES BY MAKE /* NOW IN NEW TACCLASS
WRITE MAKE MODEL YEAR /* NOW START TAC IS USED
LOOP
ON ERROR DISPLAY 'ERROR IN PROGRAM TACMULT'
END
```

The UTM TAC NAT2 has not been considered in the preceding examples; it must be defined in KDCROOT and KDCDEF.

If an explicit TACCLASS change is to take place after a WRITE, PRINT or DISPLAY statement, an EJECT must be issued before assigning the new TAC. This operation performs an unconditional output to the terminal before executing the INPUT 'TACCLASS'. Instead of the EJECT, the following statements can be used:

```
STACK TOP DATA 'A'
INPUT A(A1)
```

This sequence also performs an unconditional output to the terminal before executing the INPUT 'TACCLASS'.

Example 4:

A Natural program that allocates a UTM TAC implicitly to assign a new TAC class and then changes over to the START UTM TAC. This example uses the TAC table for synchronous transactions in the subroutine NATTAC.

```
* TACIMP1 - EXAMPLE FOR AN IMPLICIT TACCLASS SWITCH
RESET CONTROL(A3) NR(N3)
REDEFINE CONTROL (ERRFLD(A1))
INPUT 'TEST FOR AN IMPLICIT TACCLASS SWITCH' IFELD(A1)
MOVE 'S=1' TO CONTROL /* USE 1ST TAC IN TABLE --> NOTE
CALL 'NATTAC' CONTROL /* INVOKE TAC SWITCH
IF ERRFLD = 'E' DO /* ERROR CHECK
DISPLAY 'ERROR' CONTROL 'FROM NATTAC'
TERMINATE
DOEND
INPUT 'TACCLASS' /* ACTIVATE NEW TAC
READ (100) AUTOMOBILES BY MAKE /* NOW IN NEW TACCLASS
ADD 1 TO NR
WRITE NOTITLE NOHDR NR MAKE MODEL /* START TAC IS USED
LOOP
ON ERROR DISPLAY 'ERROR IN PROGRAM TACIMP1'
END
```

The value "S=1" indicates that it is a synchronous transaction and that the TAC is to be taken from the first entry in the TAC table. This is the TAC that was defined as the value of the operand of the keyword parameter TCLS1 (default value: NAT1).

The third character of the first parameter in the CALL 'NATTAC' indicates which of the four keyword parameters TCLS1 to TCLS4 applies.

Example 5:

A Natural program that allocates a UTM TAC implicitly to assign a new TAC class but does not change over to the START UTM TAC. This example uses the TAC table for synchronous transactions in the subroutine NATTAC, and processing continues with this TAC.

```
* TACIMP2 - EXAMPLE FOR AN IMPLICIT TACCLASS SWITCH
RESET CONTROL (A3) SWOFF(A1) NR(N3)
REDEFINE CONTROL (ERRFLD(A1))
MOVE 'S=4' TO CONTROL /* USE 4TH TAC IN TABLE --> NOTE
MOVE 'Y' TO SWOFF /* NO RESET TO START TAC
CALL 'NATTAC' CONTROL SWOFF /* INVOKE TAC SWITCH
IF ERRFLD = 'E' DO /* ERROR CHECK
DISPLAY 'ERROR' CONTROL 'FROM NATTAC'
TERMINATE
DOEND
INPUT 'TACCLASS' /* ACTIVATE NEW TAC
READ (100) AUTOMOBILES BY MAKE /* NOW IN NEW TACCLASS
ADD 1 TO NR
WRITE NR MAKE MODEL YEAR
LOOP
ON ERROR DISPLAY 'ERROR IN PROGRAM TACIMP2'
END
```

The value "S=4" indicates that it is a synchronous transaction and that the TAC is to be taken from the fourth entry in the TAC table. This is the TAC that was defined as the value of the operand of the keyword parameter TCLS4 (default value: NAT4).

The TAC NAT4 is not defined in the examples of KDCROOT and KDCDEF; in practice, the user must supply suitable definitions.

Using the TAC table has the advantage that the UTM TAC does not have to be coded explicitly in the Natural program. The Natural programs contain merely the relative priority "weights" of the transactions to be executed. The system administrator can allocate and change the names of the UTM TACs without having to change the Natural programs.

For testing Natural programs with TACCLASS change for synchronous transactions, please note the following: To verify correct operation of the TACCLASS change, the Natural program can be tested without the statement(s) "CALL 'NATTAC' *operand1* (*operand2*) (*operand3*)". If the INPUT 'TACCLASS' statement produces only the output 'TACCLASS' on the terminal, the program is correct. The operand(s) for the call to NATTAC must be set correctly. The UTM processing terminates with error code KM01 whenever a UTM TAC that is not defined in KDCROOT and KDCDEF is used.

Step 4: Allocation of TAC Classes for Asynchronous Transactions within one Natural/UTM Application

The TAC class for asynchronous transactions within a Natural/UTM application can be changed with the statement:

```
CALL 'NATTAC' operand1 [operand2]
```

<i>operand1</i>	Must contain the value "A= <i>n</i> ", where "A" denotes "asynchronous" and " <i>n</i> " is an integer in the range from 0 to 4 that denotes the priority level of the transaction in subroutine NATTAC's table of transaction codes for asynchronous TACs. The form of the operand is analogous to the form of the operand for synchronous transactions.
<i>operand2</i>	Optional. Contains the UTM TAC for the required TAC class if <i>operand1</i> has the value "A=0".

All UTM TACs for asynchronous transactions must begin with the character string which is defined as unique identifier for asynchronous TACs in parameter ASYNTAC of macro NATUTM. Conversely, the UTM TACs for synchronous transactions must not begin with this string.

Example 6:

A Natural program that performs initialization for asynchronous transaction processing, using the UTM TAC NATAS. This is the standard TAC for asynchronous transactions. See also the description of the keyword parameter ASYNTAC of macro NATUTM.

```
* STARTAS - EXAMPLE FOR ASYNCHRONOUS TRANSACTION WORKING
* WITHIN ONE APPLICATION - USING THE STANDARD TAC FORMAT LS=145
RESET PARM1(A144) PRDEST(A8) LTDEST(A8)
MOVE 'PRINTER1' TO PRDEST
MOVE *INITID TO LTDEST
COMPRESS 'SENDER=' PRDEST ',OUTDEST=' LTDEST ', '
'MENU=F,STACK=(LOGON APPL1;READAUTO)' INTO PARM1
LEAVING NO
CALL 'NATASYN'
SET CONTROL 'H'
WRITE NOTITLE NOHDR PARM1
INPUT 'ASYNTASK INVOKED - HOPEFULLY' IFELD(A1)
END
```

Example 7:

A Natural program that initializes asynchronous transaction processing and allocates the UTM TAC NATAS1 for assignment to another TAC class.

```
* STASTAC - EXAMPLE FOR ASYNCHRONOUS TRANSACTION WORKING
* WITHIN ONE APPLICATION
* AND SWITCH TO A NEW TACCLASS
FORMAT LS=145
RESET PARM1(A144) PRDEST(A8) LTDEST(A8) CONTROL(A3) NEWTAC(A8)
REDEFINE CONTROL (ERRFLD(A1))
MOVE 'PRINTER1' TO PRDEST
MOVE *INIT-ID TO LTDEST
COMPRESS 'SENDER=' PRDEST ',OUTDEST=' LTDEST ', '
'MENU=F,STACK=(LOGON APPL1;READAUTO)' INTO PARM1
LEAVING NO
MOVE 'A=0' TO CONTROL /* ASYNCHR. TAC, EXPLICIT --> NOTE
MOVE 'NATAS1' TO NEWTAC /* SET NEW TAC
CALL 'NATTAC' CONTROL NEWTAC /* INVOKE TAC SWITCH
IF ERRFLD = 'E' DO /* ERROR CHECK
DISPLAY 'ERROR' CONTROL 'FROM NATTAC'
TERMINATE
DOEND
CALL 'NATASYN' /* INVOKE ASYNCHRONOUS TAC
SET CONTROL 'H'
WRITE NOTITLE NOHDR PARM1
INPUT 'ASYNAC INVOKED - HOPEFULLY' IFELD(A1)
END
```

The value "A=0" indicates that it is an asynchronous transaction and that the TAC is passed explicitly in the second parameter of the CALL 'NATTAC', which means that the TAC table is not used.

```
MOVE 'A=1' TO CONTROL
CALL 'NATTAC' CONTROL
```

The procedure for using the TAC table (see keyword parameters TCLA1 - TCLA4 in the section Keyword Parameters of Macro NATUTM) corresponds to the procedure for synchronous transactions.

An example of the program that is to be executed asynchronously (READAUTO):

```
* READAUTO - ASYNCHRONOUS Natural PROGRAM
READ (75) AUTOMOBILES BY MAKE
WRITE MAKE MODEL HORSEPOWER BODY-TYPE YEAR
LOOP
ON ERROR TERMINATE
TERMINATE
END
```

Example 7 (continued):

```

*+-----+
*I PTERM 9750 DEFINITION I
*+-----+
DEFAULT PTERM PRONAM=VR , PTYPE=T9750 , TERMN=FE , CONNECT=N
PTERM DFDSS001 , LTERM=DF97501
PTERM DFDSS002 , LTERM=DF97502
PTERM DFDSS003 , LTERM=DF97503
*+-----+
*I LTERM DEFINITION I
*+-----+
DEFAULT LTERM USAGE=D , STATUS=ON , ANNOAMSG=Y , RESTART=YES
LTERM=DF97501
LTERM=DF97502
LTERM=DF97503
*+-----+
*I SFUNC DEFINITION I
*+-----+
SFUNC F1 , RET=21Z
SFUNC F2 , RET=22Z
SFUNC F3 , RET=23Z
SFUNC F4 , RET=24Z
SFUNC F5 , RET=25Z
SFUNC K1 , RET=26Z
SFUNC K2 , RET=27Z
SFUNC K3 , RET=28Z
SFUNC K4 , RET=29Z
END

```

The desired UTM TAC must always be allocated in the Natural program that initializes the asynchronous transaction processing (the use of the standard TAC for asynchronous transaction processing is an exception; see the description of the keyword parameter ASYNTAC in the macro NATUTM). The program that is to be executed asynchronously then runs in the desired TAC class. Since each asynchronous Natural program must be ended with the TERMINATE statement, the UTM DC transaction is also ended (PEND 'FI') when the program ends.

The program that initializes the asynchronous transaction processing always runs in a synchronous transaction. Thus it is feasible to perform a change of TACCLASS using the procedure for synchronous transactions. This change can take place before allocating the asynchronous TACs or after initializing the asynchronous transaction (INPUT statement).

Step 5: Assign the TAC Class for Asynchronous Transactions between two Natural/UTM Applications

It is not necessary to call NATTAC for asynchronous transaction processing between two Natural/UTM applications. The necessary UTM TAC is allocated explicitly in the Natural program; see also Asynchronous Transaction Processing.

UTM TACCLASS Switch

The following figure illustrates the logic of a UTM TACCLASS switch for synchronous transactions:

Generating a Natural/UTM Application

The following programs and macros must be assembled to generate a Natural/UTM application:

KDCROOT	UTM interface module.
NATUTM	Front-end part of the Natural UTM Interface.
BS2STUB	Common memory pool definition.
FREXIT	Format exit module (only if the default parameter is to be changed).
NURENT	Reentrant part of the Natural UTM Interface.
NTPRM	Natural parameter module.
NTSWPRM	Swap pool parameter module.

This list does not include the utility programs of the Natural UTM Interface.

The following example shows how to generate an application.

```

OPTION GEN=ALL,ROOTSRC=INPUT.KDCROOT.KDCNATP
ROOT KDCNATP
MAX APPLINAME=NATUTM,APPLIMODE=S,KDCFILE=(NATUTM,S)
MAX KB=400,SPAB=8192,NB=5120,TRMSGLTH=5120
MAX TASKS=10,ASYNTASKS=3
MAX GSSBS=0,KSSBS=1
MAX LOGACKWAIT=600,RESWAIT=(600,1200),TERMWAIT=(1800,0)
MAX PGPOOL=(88,80,95),CONRTIME=2,RECBUF=(400,2048)
MAX DPUTLIMIT1=(001,23,59,59),DPUTLIMIT2=(001,23,59,59)
MAX LPUTLTH=0
*+-----+
*I EXIT DEFINITIONS: STARTUP (CSECT NAME OF NATUTM) I
*I SHUTDOWN (CSECT NAME OF NATUTM) I
*I FORMAT (FREXIT) I
*+-----+
EXIT PROGRAM=NUSTART,USAGE=START
EXIT PROGRAM=NUSTART,USAGE=SHUT
EXIT PROGRAM=FREXIT,USAGE=FORMAT
*+-----+
*I P R O G R A M D E F I N I T I O N S I
*+-----+
DEFAULT PROGRAM COMP=ASSEMB
PROGRAM NUSTART
PROGRAM FREXIT
PROGRAM NUERROR
PROGRAM AUTOTAC
PROGRAM KDCADM,COMP=SPL4
*+-----+
*I SYNCHRONOUS TACS FOR Natural/UTM I
*I THE ERROR EXIT 'NUERROR' MUST BE DEFINED FOR EACH TAC I
*+-----+
DEFAULT TAC TYPE=D,PROGRAM=NUSTART,EXIT=NUERROR,CALL=BOTH
TAC NAT,ADMIN=NO,TIME=0
TAC AUTOCONN
*+-----+
*I BADTACS DEFINITION FOR Natural/UTM I
*I THE ERROR EXIT 'NUERROR' MUST BE DEFINED FOR EACH TAC I
*+-----+
TAC KDCBADTC,CALL=FIRST,PROGRAM=AUTOTAC,EXIT=NUERROR,TYPE=D
*+-----+

```

```

*I ASYNCHRONOUS TACS FOR Natural/UTM I
*I THE ERROR EXIT 'NUERROR' MUST BE DEFINED FOR EACH TAC I
*+-----+
DEFAULT TAC TYPE=A,PROGRAM=NUSTART,EXIT=NUERROR,CALL=FIRST
TAC NATAS
TAC NATSY
*+-----+
*I UTM ADMINISTRATOR TACS I
*+-----+
DEFAULT TAC PROGRAM=KDCADM,ADMIN=Y,TYPE=D,CALL=BOTH
TAC KDCTAC
TAC KDCLOG
TAC KDCSHUT
TAC KDCAPPL
TAC KDCINF
TAC KDCUSER
TAC KDCSEND
TAC KDCDIAG
TAC KDCLTERM
TAC KDCPTERM
TAC KDCSWTCH
TAC KDCHELP

```

See also the Siemens documentation "UTM Generierung und Administration" (UTM Generation and Administration).

Generating the Natural UTM Interface

1. The operands of the keyword parameters of macro NATUTM must be set to the correct values as required; the macro NATUTM must then be assembled.

Example of NATUTM Macro Call:

```

NUSTART NATUTM APPLNAM=NATUTM, --> Note 1 -
NUCNAME=NATvrs, --> Note 2 -
LINK=TACSWTCH --> Note 3 -
PARMOD=24, --> Note 4 -
ROLLACC=UPAM-AS, --> Note 5 -
ROLLTSZ=180, --> Note 6 -
TERMTAB=(SWP,TERMNAME), --> Note 7 -
UMODE=(S,G) --> Note 8

```

Note	
1	The CSECT name of the front-end part of the Natural UTM Interface is specified as NUSTART (default value). The name of the Natural/UTM application is specified as NATUTM.
2	The name of the link-edited reentrant part of the Natural/UTM application is specified as NATvrs; this is also the name of the common memory pool into which the reentrant part will be loaded.
3	A TABLE macro call is to be executed for program TACSWTCH. This means that this program must be linked in the front-end part of the Natural/UTM application.
4	The Natural/UTM application runs in 24-bit addressing mode.
5	The access method to the Natural roll file is specified as UPAM with P1-Eventing for asynchronous writes.
6	The maximum thread size of the Natural roll file is specified as 180 KB.
7	The internal terminal control table is allocated in the Natural swap pool; the logical terminal name will be used for identifying the entries in the terminal control table.
8	The user dialogue with Natural is to take place in "single" mode; that is, one terminal can initiate one Natural session. Messages at restart, logoff and also free-running messages (asynchronous processing) are to be output in German.

The operands of the other keyword parameters of macro NATUTM are not specified since the default values apply.

2. Assemble the macro NURENT (the reentrant part of the Natural UTM Interface). In this example, no changes are required to the keyword parameters. The CSECT name of the assembled macro NURENT is "NURENT".
3. Assemble the macro BS2STUB with the common memory pool definitions specified in macro ADDON.
4. Assemble the Natural parameter module. The sample NTPRM macro call must be adapted to suit the local environment.
5. Assemble the swap pool parameter module (macro NTSWPRM).

Linking the Non-Reentrant Front-End Part and the Reentrant Part

The front-end part and the reentrant part of the Natural UTM application can be linked using the JCL supplied. This JCL should be checked and modified as required to suit the local environment (library names, etc.) before being used. Special features in the JCL are indicated by REMARK statements.

Setting Up the Natural Roll File

The size of the Natural swap file must be calculated and the file must be allocated with link name PAMNAT.

The Start Job for a Natural UTM Application

JCL examples for starting the Natural UTM application are supplied. Before use, the JCL should be checked and modified as required (UTM startup parameters, dataset names, etc.).

Optimizing Natural UTM Applications

The following points should be considered if the performance of a Natural UTM application is unsatisfactory:

- Can poor performance be localized to one or more particular Natural programs? If so, optimize the program(s) by redesigning. These programs can be identified by using the Natural monitor in library SYSTP.
- Is the swap I/O rate too high? By using the program MENU in library SYSTP you can check how efficiently the Natural swap pool is being used. The statistical information provided about the swap pool also helps to answer the following questions:
 - Is the number of logical swap pools and their slot lengths appropriate? Function SW in the main menu of SYSTP offers various possibilities for controlling the Natural swap pool optimization.
 - Has the Natural swap pool been defined large enough? Increasing the size of the swap pool and/or generating swap pool data space reduces the swap I/O rate considerably.
- Is the Natural buffer pool too small? Information about the size and occupancy of the Natural buffer pool can be obtained with the Natural utility SYSBPM, which is described in the Natural Utilities documentation.
- Has the number of UTM tasks been chosen correctly? This is strongly dependent upon the path lengths of the individual transactions and the number of terminals connected.
- Is it possible that particular transactions (so-called "long jobs") are loading the available UTM tasks so heavily that the shorter transactions are suffering from poor throughput as a result? If this is the case, the UTM TACCLASS concept and/or the asynchronous transaction processing facilities should be used.
- Does the Natural Roll File consist of too many extents on one disk drive (physical chained I/O is not possible over extent boundaries), or is the Natural Roll File on a very heavily used disk drive? If possible, allocate the Natural Roll File to one or more lightly-used disk drives, with only one extent on each.

These suggestions should be considered in the light of the total system environment, including such factors as available storage, storage paging rates, disk and channel I/O traffic loads, etc.

Several Applications with one Common Natural

See also: Natural Shared Nucleus under BS2000/OSD (in the Natural Operations for Mainframes documentation).

To save storage space, it can be desirable for several Natural UTM applications to share a common Natural reentrant part in a common memory pool in the class 6 storage. The following steps must be taken when generating the Natural UTM application:

- The global Natural load pool must be defined with the keyword parameters of module CMPSTART, for example:

```
NAME=NATSHARE, POSI=ABOVE, ADDR=250, PFIX=YES, SIZE=2MB
LIBR=NATvrS.USER.MOD
```

For more information, see CMPSTART Program (in the Natural Operations for Mainframes documentation).

Notes:

- NATSHARE is the name of the linked Natural reentrant part. It is also the name of the common memory pool.
- The operand of parameter PFIX must be YES.
- The operand of parameter ADDR must be defined.
- The operand of parameter LIBR must contain the name of the module library from which the Natural reentrant part is to be loaded.
- The reentrant part of the Natural UTM driver (the assembled module of macro NURENT) must be linked to the front-end part of several applications.
- The operand of keyword parameter NUCNAME must be defined for each assembly of macro NATUTM as the same (in this example: NUCNAME=NATSHARE).
- The definition of the Natural load pool in the ADDON macro for the assembly of macro BS2STUB must be the same for all applications, for example:

```
STUBSHAR BS2STUB PARMOD=31, PROGMOD=ANY
ADDON NAME=NATSHARE, STAT=GLOBAL
```

For more information, see ADDON Macro in the Natural Operations for Mainframes documentation.

Lists of Shared and Application-Specific Parameter Modules

If application-specific Natural parameter modules are to be used, they must be linked to the front-end parts of the Natural UTM applications, which means that there is a parameter module in each front-end part. This also applies to the swap pool parameter module.

Only the addresses defined in the CSTATIC list of the parameter module of the front-end part are considered; if any of these addresses cannot be resolved in the front-end part (because they refer to the reentrant part), Natural tries to resolve these addresses with the CSTATIC list in the parameter module of the reentrant part. Thus it is allowed to have unresolved CSTATIC addresses when linking the front-end part, provided they can be resolved by the reentrant part.

As the CSTATIC list of the reentrant part is only used for those addresses which cannot be resolved by the front-end part, *all* CSTATIC entries to be used (whether they are in the front-end part or in the reentrant part) must be defined in the CSTATIC list of the parameter module of the front-end part.

Entering and Defining Dynamic Natural Parameters

The following possibilities exist for entering and defining the Natural dynamic parameters:

- entering the dynamic parameters together with the UTM TAC when logging on to the application;
- passing the dynamic parameters from another UTM partial program using "MPUT" and "PEND PR(OGRAM)";
- defining the dynamic parameters in the operand of the keyword parameter MSPAR1. They then apply to all users of this application and cannot be changed.

UTM User Restart

When a Natural session is started, any Natural dynamic parameters defined are saved up to a length which is defined in the operand of keyword parameter SVDYPRM in macro NATUTM. In case of a user restart situation, these saved data are automatically reused when the Natural session is started again. This also applies when the start of the Natural session results from a PEND PR(OGRAM) of another UTM partial program.

See also Global (Restartable) Swap Pool in the Natural Operations for Mainframes documentation.

Adabas Priority Control

Adabas priority control has no connection with the priority control of BS2000/OSD. Unlike with BS2000/OSD priority control, for Adabas a higher priority value means higher priority. If several requests are in the Adabas command queue at the same time, the request with the highest priority is processed first by Adabas and "1" is added to the priority of the other requests that are in the command queue at this time.

Under certain conditions, it may be useful to assign to the Adabas task a lower BS2000/OSD priority than to the UTM tasks.

The following keyword parameters in macro NATUTM can be used to control Adabas priority control for UTM transactions:

ADAPRI	Activation of Adabas priority control for UTM transactions.
APRISTD	Assignment of standard Adabas priority for all UTM transactions to which no priority is assigned individually.
TCLSn	Assignment of Adabas priority for individual synchronous UTM transactions.
TCLAn	Assignment of Adabas priority for individual asynchronous UTM transactions.

If Adabas priority control is activated for UTM transactions (parameter ADAPRI=YES), it is also in effect for non-Natural programs which access Adabas via the subroutine ADACALL; see the keyword parameter ADACALL in the macro NATUTM.

By defining different Adabas priorities for different transactions with the above parameters, and at the same time using the UTM TACCLASS concept, it is possible to set up a very sophisticated system of priority control. However, when you explicitly assign Adabas priorities to UTM transaction, you should take into consideration the standard priorities Adabas assigns to other processes (for example, TIAM or batch processing).