# Natural Development Server

## Natural Development Server for z/OS (Com-plete)

Version 9.1.2 for z/OS (Com-plete)

October 2020

ADABAS & NATURAL

This document applies to Natural Development Server Version 9.1.2 for z/OS (Com-plete) and all subsequent releases.

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

Copyright © 2001-2020 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at http://softwareag.com/licenses.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at http://softwareag.com/licenses/ and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or re-strictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at http://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

**Document ID: NDV-NNDVDOC-912-20201002**

# Table of Contents


Preface ..... v
1 About this Documentation ..... 1
Document Conventions ..... 2
Online Information and Support ..... 2
Data Protection ..... 3
2 Introducing Natural Development Server ..... 5
Purpose of Natural Development Server ..... 6
Remote Development Functions ..... 6
3 Development Server File ..... 9
Purpose of the Development Server File ..... 10
Relations between FDIC and the Development Server File ..... 10
Unique Development Server File ..... 10
4 Natural Development Server on Mainframes ..... 13
Development Server Concept ..... 14
Front-End Stub NATRDEVS ..... 15
Front-End ..... 16
Transaction Processors ..... 16
Gateway Module ..... 17
Server Monitor ..... 17
Product Interaction ..... 17
5 Prerequisites ..... 19
General Prerequisites for NDV Installation ..... 20
Prerequisites for NDV for z/OS (Com-plete) ..... 21
6 Installing the Natural Development Server on z/OS (Com-plete) ..... 23
Prerequisites ..... 24
Content of the Development Server Distribution Medium ..... 24
Installation Procedure ..... 24
7 Configuring the Natural Development Server ..... 29
Configuration Requirements ..... 30
NDV Configuration File ..... 31
NDV Configuration Parameters ..... 32
NDV Configuration File Example ..... 42
NDV Server Datasets ..... 42
NDV User Exits (Coded in Natural) ..... 42
Other NDV User Exits ..... 43
8 Operating the Natural Development Server ..... 45
Starting the Natural Development Server ..... 46
Terminating the Natural Development Server ..... 46
Monitoring the Natural Development Server ..... 47
Runtime Trace Facility ..... 48
Trace Filter ..... 49
9 Monitor Client NATMOPI ..... 51
Introduction ..... 52

Command Interface Syntax ..... 52
Command Options Available ..... 52
Monitor Commands ..... 53
Directory Commands ..... 53
Command Examples ..... 54
10 HTML Monitor Client ..... 55
Introduction ..... 56
Prerequisites for HTML Monitor Client ..... 56
Server List ..... 56
Server Monitor ..... 58
11 SPoD-Specific Limitations and Considerations ..... 61
Limitations ..... 62
Performance Considerations ..... 71
Natural Documentation and Online Help ..... 75
12 Natural Development Server Frequently Asked Questions ..... 77
Natural Development Server starts and terminates immediately ..... 78
Which dataset should I analyze to get error information? ..... 78
Trace output shows: Cannot load Natural front-end ... ..... 79
Trace output shows: Transport initialization failed, EDC8115I address already in use ..... 79
Trace output shows: Error at: Template runtime connect ..... 79
Definitions required in Natural Security ..... 80
Map Environment fails with a NAT3048 ..... 80
Map Environment fails with Stub RCnn ..... 80
Special characters are not translated correctly ..... 82
Characters are not displayed correctly in the terminal emulation of Natural Studio ..... 84
How do I find out which hexadecimal value must be specified for TABA1/TABA2? ..... 84
The modifications of TABA1/TABA2 do not apply to sources listed in the remote debugger ..... 85
Accessing work files ..... 85
Are there any Natural profile parameter settings required for NDV? ..... 85
NAT9915 GETMAIN for thread storage failed ..... 85
The NDV server consumes a lot of CPU time even if only a few clients are using it ..... 86
I get a NAT0873 internal error at user authentication for Map Environment ..... 86
The server fails to start with return code 4 and in the error log I find ‘Transport initialization failed‘ ..... 86

# Preface

This documentation applies to Natural Development Server (product code NDV) for z/OS (Com-plete).

Natural Development Server for z/OS is released together with Natural for Mainframes. It has the same version number as Natural for Mainframes.

For information on changes, enhancements or new features in this version of Natural Development Server, see the *Release Notes* in the corresponding Natural for Mainframes documentation.

| | |
|---|---|
| **Introducing Natural Development Server** | Describes purpose and functionality of Natural Development Server which is used in conjunction with NaturalONE or Natural for Windows (as a client) in a Natural Single Point of Development (SPoD) environment. |
| **Development Server File** | Describes purpose and use of the Natural Development Server file, a central dictionary file that is structurally identical to the Natural system file `FDIC`. |
| **Natural Development Server on Mainframes** | Describes concept and architecture of Natural Development Server. |
| **Prerequisites** | Describes prerequisites that apply when you install Natural Development Server on a mainframe computer. |
| **Installing the Natural Development Server** | How to install Natural Development Server under the operating system z/OS (Com-plete). |
| **Configuring the Natural Development Server** | How to configure Natural Development Server. |
| **Operating the Natural Development Server** | How to operate Natural Development Server. |
| **Monitor Client NATMOPI** | Describes the Monitor Client NATMOPI, a character-based command interface for monitoring the various types of servers that are provided in a mainframe Natural environment. |
| **HTML Monitor Client** | Describes the HTML Monitor Client, a monitor interface that supports any web browser as a user interface for monitoring the various types of servers that are provided in a mainframe Natural environment. |
| **SPoD-Specific Limitations and Considerations** | Describes the limitations which are due to the different capabilities of the graphical user interface available on the local site and the character-based user interface that exists on the remote site. In addition, this document includes hints which are important for the efficient use of the remote development facilities. |
| **Natural Development Server Frequently Asked Questions** | Contains frequently asked questions concerning Natural Development Server. |

**Related Documentation**

- Natural Single Point of Development documentation
- NaturalONE documentation
- Natural for Windows documentation
- Natural for Mainframes documentation

# 1 About this Documentation


- Document Conventions ..... 2
- Online Information and Support ..... 2
- Data Protection ..... 3

## Document Conventions

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

## Online Information and Support

### Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

### Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.aspx** and give us a call.

**Software AG TECHcommunity**

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

## Data Protection

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

# 2 Introducing Natural Development Server


- Purpose of Natural Development Server ..... 6
- Remote Development Functions ..... 6

This chapter describes the purpose and the functions of Natural Development Server (product code NDV) which are used in conjunction with NaturalONE or Natural for Windows (as client) in a Natural Single Point of Development (SPoD) environment.

## Purpose of Natural Development Server

Natural Development Server enables you to use NaturalONE or the Natural Studio development environment provided by Natural for Windows to develop and test Natural applications in a remote Natural environment running under the operating system z/OS (Com-plete).

For more information on NaturalONE and remote development, see

- NaturalONE documentation (describes the SPoD client side; how to manage offloaded Natural objects in the Eclipse workspace, and also how to modify them directly on a development server).
- Natural Single Point of Development documentation (general information).

For more information on Natural Studio and remote development, see

- Natural for Windows documentation or Help system (describes the SPoD client side; how to manage Natural objects directly on a development server).
- Natural Single Point of Development documentation (general information).

## Remote Development Functions


- Establishing a Connection between Client and Server
- Using the Remote Development Functionality


### Establishing a Connection between Client and Server

A connection to an active development server can be established by mapping it in the client (that is, in NaturalONE or Natural Studio). A dialog will be shown for setting up the connection in which you have to specify the following information:

### Server

| | |
|---|---|
| **Host name** | The host name defines the remote node name where the server is running (or the IP address of the server). |
| **Server port** | The server port defines the TCP/IP port number for the development server. |
| **Environment name** | The environment name can be used to give the addressed server a logical (descriptive) name. If this box is left blank, a default name will be created automatically. |

### Startup

| | |
|---|---|
| **Session parameters** | If dynamic parameters are required for your development server, specify them in this text box. Otherwise, leave this text box blank.<br><br>**Note:** Specifying a parameter string in this text box causes the profile specification of the NDV configuration parameter `DEFAULT_PROFILE` to be overwritten. |
| **User ID** | Your user ID is automatically provided. |
| **Password** | If Natural Security is installed on the development server, specify the required password in this text box. Otherwise, leave this text box blank. |

These settings are transferred to the selected Natural Development Server and evaluated to create an exclusive Natural session that is responsible for executing all development requests for that environment. Once you have sucessfully mapped a development server, the Natural objects of the connected remote development environment are shown in NaturalONE or Natural Studio.

## Using the Remote Development Functionality

You can use the entire functionality of NaturalONE or Natural Studio to create, edit, stow or execute Natural objects on the remote Natural environment. You can map to multiple environments from one NaturalONE or Natural Studio. Each mapped environment owns a Natural session on the Natural Development Server, even if you map multiple environments on the same server.

When you are working with NaturalONE, it is recommended that you work in the so-called "local mode". In local mode, the sources are no longer stored or modified directly on the development server. The central place for keeping the sources is now the Eclipse workspace which is connected to a version control system.

# 3 Development Server File


- Purpose of the Development Server File ..... 10
- Relations between FDIC and the Development Server File ..... 10
- Unique Development Server File ..... 10

This chapter describes purpose and use of the Natural Development Server file, a central dictionary file that is structurally identical to the Natural system file FDIC.

## Purpose of the Development Server File

As Natural stores its data in system files, Natural Development Server stores its data in the system file that is assigned to the Natural parameter FDIC, a logical system file which is called the "development server file".

The development server file is used as a central dictionary file for storing Natural applications and the links to objects making up an application. It also holds object locking information. This information is not bound to certain groups of application developers, but has an impact on the entire application development of an enterprise. Therefore, this file should be available only once, to ensure that the application definitions and locking states are kept consistent.

## Relations between FDIC and the Development Server File

The development server file layout corresponds to the file layout of the Natural system file FDIC used by Predict. This means that the central dictionary file can also be used to hold Predict data, but Predict is not a prerequisite for using the development server file. This enables you to use your existing application documentation in the application definitions of the remote development environment.

## Unique Development Server File

It is of vital importance that the various remote development environments that can be mapped use a common and unique development server file.

Non-compliance with this requirement may give rise to inconsistencies in object locking and in the applications existing in the application workspace.

**NTDYNP Macro**

To prevent the FDIC parameter from being overwritten when a Natural Development Server is mapped, you are strongly recommended to prevent the NTDYNP macro from being used to specify FDIC as a dynamic parameter.

**Under Natural Security**

In a Natural Development Server that is protected by Natural Security, the use of another `FDIC` file in the application workspace is prevented if the application security profiles are activated. See also *Application Protection* in the *Natural Security* documentation.

# 4 Natural Development Server on Mainframes


- Development Server Concept ..... 14
- Front-End Stub NATRDEVS ..... 15
- Front-End ..... 16
- Transaction Processors ..... 16
- Gateway Module ..... 17
- Server Monitor ..... 17
- Product Interaction ..... 17

This chapter describes the concept and the architecture of the Natural Development Server (product code NDV) which is designed for use under z/OS (Com-plete).

## Development Server Concept

A Natural Development Server is a multi-user, multi-tasking application. It can host Natural sessions for multiple users and execute their requests concurrently.

The concept is based on the "serverized" Natural runtime system. Its architecture comprises a server front-end stub (development server stub `NATRDEVS`) that uses the Natural front-end to dispatch Natural sessions and to execute functionality within these sessions.

The Natural remote development server architecture basically consists of:

- **Front-end stub**

  The stub `NATRDEVS` is launched to initialize a Natural Development Server. It listens for incoming transactions and dispatches the appropriate Natural session for executing the transaction.
- **Front-end**
- **Gateway module**

  The module `NATGWSTG` provides for interaction between the Natural runtime system and the front-end stub. `NATGWSTG` is already included in the Natural nucleus and is called by the Natural runtime system to exchange the necessary request data.
- **Transaction processors**

  Transaction processors are called by the front-end stub. The application logic of each individual transaction is implemented within a transaction processor.
- Natural Driver

  Natural is driven by the Natural Com-plete interface `NCF-SERV`.
- **Server monitor**

  A monitor task allows the administrator to control the server activities, to cancel particular user sessions or to terminate the entire server, etc.

# Front-End Stub NATRDEVS

The multi-user, multi-tasking, front-end stub `NATRDEVS` is launched to initialize a Natural Development Server.


- Stub Description
- Natural System Variables Used
- Natural I/O Handling


## Stub Description

The task executing the server initialization (`TMain`) basically is the main listener which waits for incoming requests from the remote development client (Natural Studio). It owns a session directory to manage multiple clients (users) and their corresponding remote Natural sessions. `TMain` has the task to accept all incoming requests and to dispatch them to other subtasks (`TWork`). The process is as follows:

- First, a **Map Environment** command issued by the user on the client side (in the **Tools** menu of Natural Studio) connects to `TMain` to establish a connection.
- Next, `TMain` inserts the client into its session directory, attaches a new `TWork` subtask and passes the connection to TWork.
- `TWork` processes the request (indeed initializes a new Natural session if the client sends a `CONNECT` request) and replies to the client.
- After the reply, `TWork` listens on that connection for successive requests of that particular client. `TWork` remains active until the user on the client (Natural Studio) side switches the focus to a different environment (the local or a different mapped environment).
- If the user activates the environment again, `TMain` launches a new `TWork` subtask that resumes the existing Natural session from the previous TWork.

That is, each client owns one subtask `TWork` on the Natural Development Server and multiple remote Natural sessions (one for each mapped environment). This subtask remains active as long as the mapped environment on Natural Studio is the currently active environment. Each remote Natural session remains active until the user disconnects/unmaps the corresponding environment on the client side. Consequently, a Natural session can be executed under different subtasks if the user switches among multiple environments.

### Natural System Variables Used

Within a Natural Development Server session, the following Natural system variables are used:

- *TPSYS contains SERVSTUB,
- *DEVICE contains VIDEO,
- *SERVER-TYPE contains DEVELOP.

### Natural I/O Handling

The Natural runtime system allows I/O execution in the same way as in an online environment:

- A Natural Development Server intercepts the I/O and sends the 3270 data stream to Natural Studio.
- Natural Studio internally starts a terminal emulation window and passes the 3270 stream to that window.
- After I/O execution, the I/O data is sent back to the server.
- The front-end stub invokes the front-end to continue processing after I/O.

## Front-End

The Natural front-end required for a Natural Development Server is the Natural Com-plete driver NCFNUC that is delivered with the corresponding Natural Version for Mainframes.

## Transaction Processors

The transaction processors are Natural programs in the library SYSLIB that process transactions (for example, "save source", "get library list") requested by the remote development client. The transaction processors are invoked by the front-end stub.

## Gateway Module

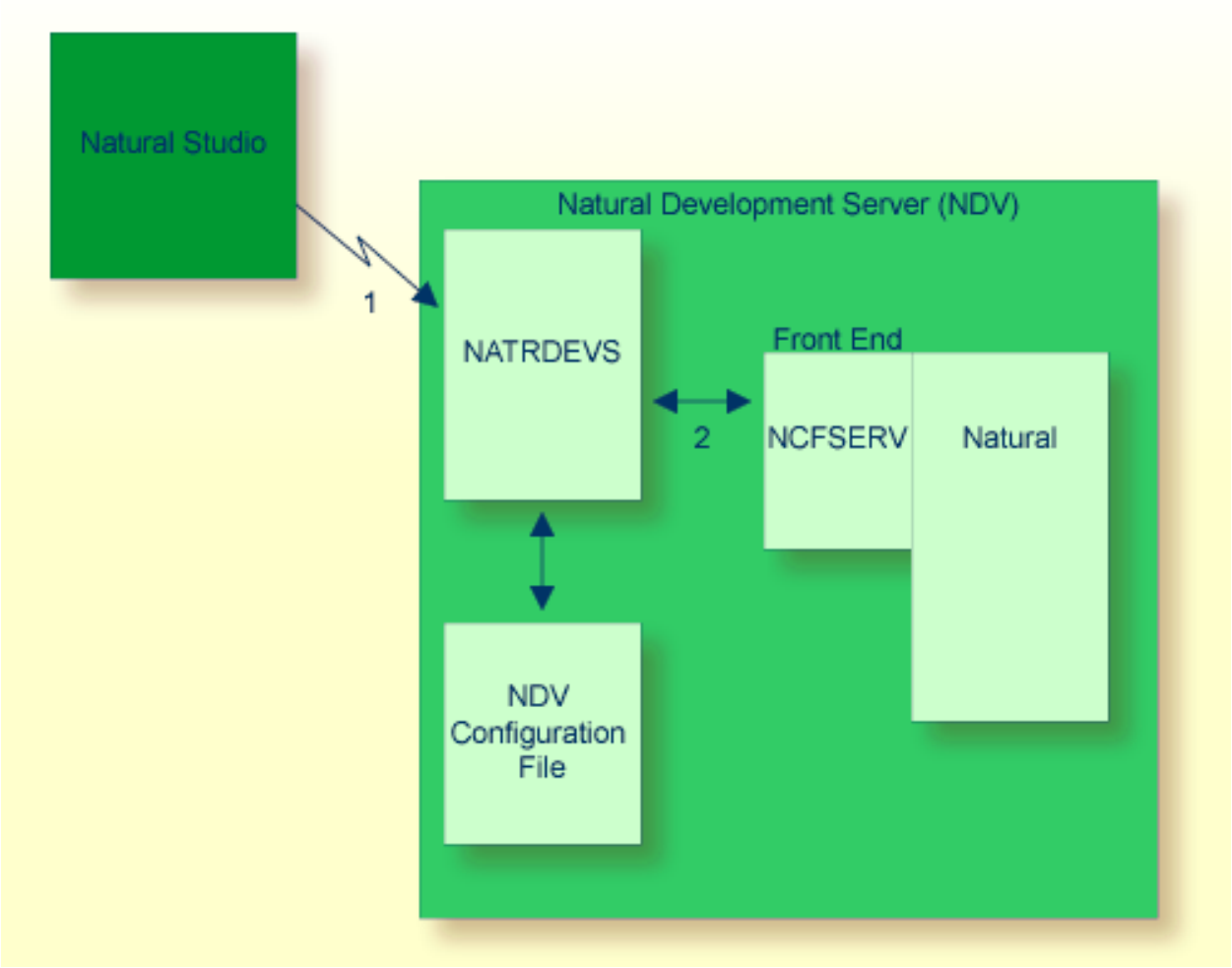
The gateway module `NATGWSTG` is already included in the Natural nucleus.

## Server Monitor

To enable the administrator to monitor the status of the Natural Development Server, a monitor task is provided which is initialized automatically at server startup. Using the **monitor commands**, the administrator can control the server activities, cancel particular user sessions, terminate the entire server, etc. See *Operating the Development Server*.

## Product Interaction

The following figure illustrates the interaction of Natural Studio used as a remote development client with a Natural Development Server.




1. Natural Studio (the client) sends a remote development request to the Natural Development Server (NDV) using the port number specified with the NDV configuration parameter `PORT_NUMBER`.
2. The Natural Development Server dispatches the Natural session using the Natural front-end you have specified with the NDV configuration parameter `FRONTEND_NAME` (`NCFSERV` in this example).

# 5 Prerequisites


- General Prerequisites for NDV Installation ..... 20
- Prerequisites for NDV for z/OS (Com-plete) ..... 21

This chapter describes the prerequisites that apply when you install a Natural Development Server (product code NDV) on a mainframe computer.

## General Prerequisites for NDV Installation

- The currently applicable version of Natural for Mainframes must be installed; refer to Empower at *https://empower.softwareag.com/*.

  **Important:** Any user-written exits not written in Natural and used within a Natural Development Server environment must be reentrant and thread-safe (capable to run in a multi-tasking environment).

- If you are using Predict and you have to migrate to a Predict version specified under *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes*, you are strongly recommended to migrate to the newer Predict version *before* you install the Natural Development Server.

  **Important:** If you do not migrate to a Predict version specified under *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes* before starting the Natural Development Server installation, you will have to define a new Natural system file `FNAT` and a new Development Server File (`FDIC`). The current version of Natural for Mainframes and the desired additional products must have been loaded on the Natural system file `FNAT` before you start the installation of the Natural Development Server.

- To use other Software AG products in conjunction with the Natural Development Server, refer to the section *Software AG Product Versions Required with Natural* in the current *Natural Release Notes for Mainframes* for the required version.
- The Software AG Editor must be installed. You are recommended to set the size of the editor buffer pool to 1024 KB.

  If you are using System Maintenance Aid (SMA), the necessary modules are linked when the SMA parameter `SAG-EDITOR` is set to `Y` (Yes). This is the default.

  If you are installing without SMA, see Installation for z/OS, *Installing Software AG Editor*.
- The prerequisites for the operation of a remote development client must be fulfilled in addition. Natural for Windows or NaturalONE must have been installed on the PC client. For information on the applicable version of Natural for Windows, refer to Empower at *https://empower.softwareag.com/*.
- Natural Development Server Version 9.1 is compatible with all supported versions of Natural for Mainframes, Natural for Windows and NaturalONE.

**Note:** For information about plug-ins and add-on products available, refer to Empower at *https://empower.softwareag.com/* and the section *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes*.

## Prerequisites for NDV for z/OS (Com-plete)

In addition to the **general prerequisites** described above, the following operating-system-specific prerequisites apply:

- z/OS must be installed.

  Version as specified under *Operating/Teleprocessing Systems Required* in the current *Natural Release Notes for Mainframes*.
- Com-plete must be installed (product code COM).
- The Natural Com-plete/SMARTS Interface must be installed (product code NCF).
- To prevent the formation of endless loops in user programs running under NDV, specify a reasonable value for the CPU time limit in the SMARTS parameter `THREAD-GROUP`.
- As a prerequisite for using the client impersonation feature (parameter `SECURITY_MODE`), Natural Security must be installed.

# 6 Installing the Natural Development Server on z/OS (Com-plete)


- Prerequisites ..... 24
- Content of the Development Server Distribution Medium ..... 24
- Installation Procedure ..... 24

This chapter describes how to install a Natural Development Server (product code: NDV) under Com-plete on z/OS.

## Prerequisites

For details, refer to the section ***Prerequisites***.

## Content of the Development Server Distribution Medium

The installation medium contains the datasets listed in the table below. The sequence of the datasets and the number of library blocks needed are shown in the *Software AG Product Delivery Report*, which accompanies the installation medium.

| Dataset Name | Contents |
|---|---|
| NDV*vrs*.OBJS | Contains the object modules of the development server. See ***Natural Development Server on Mainframes***. |
| NDV*vrs*.SYSF | Contains the FDT of the Development Server File (the layout is identical with PRD*vrs*.SYSF provided with a Predict version as specified under *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes*). |
| NDV*vrs*.JOBS | Example installation jobs. |

The notation *vrs* in the dataset names and in the program samples (below) represents the version, release and system maintenance level of the product.

For the currently applicable versions refer to Empower at ***https://empower.softwareag.com/***.

## Installation Procedure

### Step 1: Allocate the development server LOAD library for Com-plete

(Job I008, Step 8420)

### Step 2: Create member NDVCONFG for Com-plete

(Job I009, Step 8410)

Step 8410 creates a sample `NDVCONFG` for the batch server.

The following parameters of the configuration file must be defined. For the other parameters, the default values may be used:

| | |
|---|---|
| `FRONTEND_NAME` | Specify the name of the NCF front-end module of your Natural for Com-plete installation. |
| `PORT_NUMBER` | Specify the TCP/IP port number under which the server can be connected. |

For a description of the parameters, refer to *Configuring the Natural Development Server*.

### Step 3: Load FDIC system file

(Job I050, Step 8403)

If you do not use Predict at all or if you have not yet migrated to a Predict version as specified under *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes*, create the development server file, using the dataset `NDV`*`vrs`*`.SYSF`.

The layout of the **Development Server File** corresponds to the layout of the Predict dictionary file.

**Note:** If you have a Predict version installed as specified under *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes*, you can ignore this step.

### Step 4: Link the object modules into the NDV load library

(Job I054, Step 8430)

See sample job `NDVI054X` on dataset `NDV`*`vrs`*`.JOBS`.

The NDV object modules must be linked with the necessary runtime extensions of your Com-plete installations into executable load modules.

The following ULIB CAT entries are necessary:

| | | |
|---|---|---|
| `PAENSTRT` | 128 KB | `Privileged=Y` |
| `NATRDEVS` | 128 KB | `Privileged=Y` |
| `NATMOPI` | 128 KB | `Privileged=Y` |

**Note:** The link jobs for NDV in z/OS Smarts/Complete environments use the same names as in a z/OS LE environment. In oder to avoid overriding of different environments in z/OS (LE and Smarts/Complete), the linkage libraries should be unique.

### Step 5: Copy DDMs and processing rules to FDIC

If you use a Predict system file FDIC as Development Server File (`FDIC`), ignore this step.

If a Predict version as specified under *Natural and Other Software AG Products* in the current *Natural Release Notes for Mainframes* has not been installed or if you do not use a Predict system file `FDIC` as Development Server File (`FDIC`), you have to copy the existing DDMs and processing rules to the Development Server File (`FDIC`), using the copy function of the Natural utility `SYSMAIN`.

### Step 6: Extend your Com-plete startup job by NDV-specific definitions

See sample job `COMJCL` on dataset `COM`*`vrs`*`.JOBS` or SMA Com-plete startup Job I025, Step 3350.

Described in the section ***Configuring the Natural Development Server***.

**z/OS Sample:**

```
//COMPLETE PROC PREFIX='COM',

...

//STEPLIB  DD ...

//COMPLIB  DD ...

//         DD NDV load library must be part of the COMPLIB

...

//NDVvrsE   DD SYSOUT=X                                    The error output for ↩
server id NDVvrs

//NDVvrsC   DD DISP=SHR,DSN=PREFIX.NDV.CONFIG(NDV NDVCONFG) The configuration file ↩
for server id NDVvrs

//NDVvrsT   DD SYSOUT=X                                   The STDOUT output for ↩
server id NDVvrs   0

...
```

The sample above is not a complete startup job but just an excerpt containing the NDV relevant information.

For the necessary `SYSPARM` adaptations please refer to ***SYSPARM Example for the Natural Development Server*** in ***Configuring the Development Server***.

### Step 7: Debug a Natural batch job with Debug Attach for NaturalONE

(Job I200, Step 8418)

**Sample JCL:**

```
//NATEX    EXEC PGM=NATvrsBA,REGION=8M
//STEPLIB  DD DSN=SAGLIB.NDVvrs.LOAD,DISP=SHR
//         DD DSN=SAGLIB.SMALOAD,DISP=SHR
//         DD DSN=SAGLIB.ADAvrs.LOAD,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//*
//CMPRMIN  DD *
DBGAT=(ACTIVE=ON,CLID=CLIENT,HOST=DASSERV,PORT=2500)
RCA=(NATATDBG) RCALIAS=(NATATDBG,NATADvrs)
//*
//DDCARD   DD *
 ADARUN DB=001,DE=3390,SVC=249
//*
//SYSPRINT DD SYSOUT=*
//CMPRINT  DD SYSOUT=*
//CMSYNIN  DD *
MYLIB,DBA,DBA
MYPROG
FIN
/*
```

### Step 8: NDV clients must be defined to Natural Security

If Natural Security (NSC) is installed:

- The NDV initial user ID (default ID is `STARGATE`) must be defined in Natural Security with a valid default library. Refer also to NDV configuration parameter `INITIAL_USERID` in the section *Configuring the Natural Development Server*. Alternatively, you can specify the Natural profile parameter `AUTO=OFF` (automatic logon) for NDV.
- Each client user ID must be defined in Natural Security.

If the NDV initial user ID is not defined, the NDV server initialization aborts with a NAT0856 error message.

If an NDV client is not defined, the **Map Environment** dialog returns an NSC error.

If you logon to the server from an NDV client, make sure that the user who is defined in Natural Security has a default library or a private library defined. Otherwise, the error message NAT0815 will be displayed.

### Step 9: NDV clients must be defined to the server host

If you configure the NDV server to use an external security system (see NDV configuration parameter `SECURITY_MODE`), the NDV clients must be defined to the external security system.

# 7 Configuring the Natural Development Server


- Configuration Requirements ..... 30
- NDV Configuration File ..... 31
- NDV Configuration Parameters ..... 32
- NDV Configuration File Example ..... 42
- NDV Server Datasets ..... 42
- NDV User Exits (Coded in Natural) ..... 42
- Other NDV User Exits ..... 43

This chapter describes how to configure a Natural Development Server for z/OS (Com-plete).

# Configuration Requirements


- SMARTS SYSPARM Parameters
- SYSPARM Example for the Natural Development Server


## SMARTS SYSPARM Parameters

The Natural Development Server requires the following SMARTS `SYSPARM` parameters:

| Parameter | Definition | |
|---|---|---|
| `RESIDENTPAGE` | The following members must be defined in the SMARTS resident area:<br><br>`NATRDEVS`, `NATMONI`, `NATDSSEC`, Natural front-end (`NCFNUC`) and Natural nucleus (if you run using a split nucleus). | |
| `SECSYS` | The installed external security system (`RACF \| ACF2 \| TOPSECRET`). | |
| `SERVER` | The following `SERVER` definitions are required for the Natural Development Server: | |
| | `SERVER=(OPERATOR,TLINOPER,TLSPOPER)` | The Operator Communications Server. |
| | `SERVER=(POSIX,PAENKERN)` | The POSIX Server. |
| | The Natural local buffer pool definition. | For details, refer to the Natural Com-plete interface documentation. |
| `CDI_DRIVER` | `CDI_DRIVER=('TCPIP,PAAOSOCK')` | The SMARTS TCPIP Socket Driver for TCP/IP stack on z/OS (Com-plete). |
| `THSIZEABOVE` | `THSIZEABOVE=1024` | The storage above 16 MB that is available for each Natural Development Server subtask. This size must be large enough to keep the Natural tread, heap and stack of the Natural Development Server subtasks. A certain headroom (20 % or more, depending on your environment) is recommended. If the Natural Development Server initialization fails with an error message `NAT9915 GETMAIN for thread storage failed`, this parameter must be increased. |
| `ADASVC` | `ADASVC=`*nnn* | The Adabas SVC number of your Adabas installation. |

You can set the SMARTS `SYSPARM` parameters in the file referred by DD-name `SYSPARM` within your Com-plete startup JCL.

### SYSPARM Example for the Natural Development Server

In the following example, the notation *vrs* or *vr* stands for the relevant version, release, system maintenance level numbers. The currently applicable product version must be installed; refer to Empower at ***https://empower.softwareag.com/***.

```
* ------------------ NDV Server to launch at startup ----------*
* STARTUPPGM='NATRDEVS NDVvr'
*
*
TASK-GROUP=(DEFAULT,6)
THREAD-GROUP=(DEFAULT,(DEFAULT,252,06,15,28,N))
*
THSIZEABOVE=1024
*
SERVER=(NCFNATvr,NCFNATvr)
*
CDI_DRIVER=('TCPIP,PAAOSOCK,ADDRSPCE=...')
*
RESIDENTPAGE=NATRDEVS
RESIDENTPAGE=NDVNCFvr
RESIDENTPAGE=NATNUCvr
RESIDENTPAGE=NATHTMON
RESIDENTPAGE=NATMONI
RESIDENTPAGE=NATDSSEC
```

`ADDRSPCE` should contain the name of the TCP/IP address space.

## NDV Configuration File

A configuration file is allocated to the name `<serverid>C` (for example, ) or `STGCONFG` alternatively.

The configuration file is a text file located on a dataset under z/OS.

The configuration file contains the server configuration parameters in the form of a *keyword=value* syntax. In addition, it may contain comments whose beginning is marked with a hash symbol (#).

See also the ***NDV Configuration File Example*** shown below.

## NDV Configuration Parameters

The following NDV configuration parameters are available:


- DBG_CODEPAGE
- DEFAULT_PROFILE
- FORCE_IPV4
- FRONTEND_NAME
- HANDLE_ABEND
- HOST_NAME
- HTPMON_ADMIN_PSW
- HTPMON_PORT
- HOST_NAME
- INITIAL_USERID
- MINIMUM_STUDIO_VERSION
- O4I
- PASSWORD_MIXEDCASE
- PORT_NUMBER
- SECURITY_MODE
- SESSION_PARAMETER
- SESSION_PARAMETER_MIXED_CASE
- SESSION_TIMEOUT
- TERMINAL_EMULATION
- TRACE_FILTER
- TRACE_LEVEL
- UNICODE_SOURCE
- UPPERCASE_SYSTEMMESSAGES


### DBG_CODEPAGE

This optional configuration parameter specifies the translation table to be used by the remote debugger. By default, the remote debugger uses the code page `IBM-1047`, whereas the Natural Development Server uses `TABA1/2`.

| Value | Explanation |
|---|---|
| `USER` | Use the Natural translation tables `TABA1/2`. |

No default value is provided.

Example:

```
DBG_CODEPAGE=USER
```

### DEFAULT_PROFILE

This optional configuration parameter defines a default profile.

| Value | Explanation |
|---|---|
| *string* | The following syntax applies:<br>*profile-name,dbid,fnr,password,cipher-code*<br>**Note:** Specifying a parameter string in the **Session Parameters** text box of the **Map Environment** dialog box in Natural Studio overwrites this default profile. |

No default value is provided.

Example:

```
DEFAULT_PROFILE=RDEVS,10,930
```

The setting in the example defines that, if no parameters are defined in the **Map Environment** dialog box of Natural Studio, the session is started with the Natural profile parameter `PROFILE=(RDEVS,10,930)`.

Related parameter: `SESSION_PARAMETER`.

### FORCE_IPV4

This configuration parameter applies to z/OS and z/VSE.

This parameter allows you to enforce the use of communication method IPV4.

| Value | Explanation |
|---|---|
| YES | Enforce the use of communication method IPV4. |
| NO | First try communication method IPV6. If this fails give an error message and use communication method IPV4. This is the default value. |

### FRONTEND_NAME

This configuration parameter specifies the name of the Natural front-end to be used to start a Natural session. The front-end resides on a PDS member.

| Value | Explanation |
|---|---|
| *frontend-name* | Natural front-end to be used. Maximum length: 8 characters. |

No default value is provided.

Example:

```
FRONTEND_NAME=NAT vrs SV
```

- where *vrs* stands for the version, release, system maintenance number.

### HANDLE_ABEND

If an abend occurs in the server processing outside the Natural processing the abend is not trapped by the Natural abend handling. For this reason the NDV server has its own abend recovery.

It is recommended that you leave this parameter on its default value in order to limit the impact of an abend to a single user. If you set the value of this parameter to `NO`, any abend in the server processing terminates the complete server processing. That is, it affects all users running on that server.

| Value | Explanation |
|---|---|
| `YES` | Trap abends in the server processing, write a snap dump and abort the affected user. This is the default value. |
| `NO` | Suspend the server abend handling. |

Example:

```
HANDLE_ABEND=NO
```

### HOST_NAME

This optional configuration parameter is necessary only if the server host supports multiple TCP/IP stacks.

| Value | Explanation |
|---|---|
| *host-name* | If HOST_NAME is specified, the server listens on the particular stack specified by HOST_NAME, otherwise the server listens on all stacks. |

No default value is provided.

Example:

```
HOST_NAME=node1
```

or

```
HOST_NAME=157.189.160.55
```

### HTPMON_ADMIN_PSW

This configuration parameter defines the password required for some monitor activities (for example, Terminate Server) performed by the **HTML Monitor Client**.

| Value | Explanation |
|---|---|
| any character string | The password to be entered at the HTML Monitor Client for some monitor activities. |

No default value is provided.

Example:

```
HTPMON_ADMIN_PSW=GHAU129B
```

### HTPMON_PORT

An NDV server can be configured to host an HTTP monitor task which serves the **HTML Monitor Client** running in a web browser. It is not required to run this monitor task on each server. A single task allows you to monitor all servers running at one node.

This configuration parameter defines the TCP/IP port number under which the server monitor task can be connected from a web browser.

| Value | Explanation |
|---|---|
| 1 - 65535 | TCP/IP port number. |

No default value is provided.

Example:

```
HTPMON_PORT=3141
```

### HOST_NAME

This configuration parameter defines the host name of the NDV server.

### INITIAL_USERID

At server initialization, the Natural Development Server creates a temporary Natural session to obtain the properties of the installed Natural environment.

This configuration parameter specifies the user ID to be used for this Natural session.

| Value | Explanation |
|---|---|
| *userid* | The specified value must not exceed 8 characters, otherwise it is truncated. |
| STARGATE | This is the default value. |

Example:

```
INITIAL_USERID=NDVINITU
```

See also (in the Natural Development Server *Installation* documentation).

### MINIMUM_STUDIO_VERSION

This parameter defines a minimum version of Natural Studio which is required to operate with the NDV server. This parameter assists in performing a preliminary validation if all clients use a minimum Natural Studio version. This can be useful to smoothly upgrade to a NDV version that does not support clients whose version is below the minimum Natural Studio version.

| Value | Explanation | |
|---|---|---|
| *vvmmpp* | The Studio Version (5-6 digits), where: | |
| | *vv* | Version number (1 or 2 digits). |
| | *mm* | System maintenance level (2 digits). |
| | *pp* | Patch level (2 digits). |
| 61100 | This is the default value. | |

Example:

```
MINIMUM_STUDIO_VERSION=62100
```

### O4I

This parameter allows you to collect server data for Optimize for Infrastructure.

| Value | Explanation |
|---|---|
| YES | Collect server data for Optimize for Infrastructure. |
| NO | Do not collect server data for Optimize for Infrastructure. This is the default value. |

Example:

```
O4I=YES
```

### PASSWORD_MIXEDCASE

This parameter allows you to define whether passwords specified in the **Map Environment** dialog are translated into upper case or not.

This parameter does only apply with SECURITY_MODE=IMPERSONATE, IMPERSONATE_LOCAL or IMPERSONATE_REMOTE.

| Value | Explanation |
|---|---|
| YES | Passwords remain in mixed case. |
| NO | Passwords are translated into upper case. This is the default value. |

Example:

```
PASSWORD_MIXEDCASE=YES
```

### PORT_NUMBER

This configuration parameter defines the TCP/IP port number under which the server can be connected.

| Value | Explanation |
|---|---|
| 1 - 65535 | TCP/IP port number. |

No default value is provided.

Example:

```
PORT_NUMBER=3140
```

### SECURITY_MODE

The Natural Development Server offers a security concept that also covers the operating system resources. The client credentials are validated at the operating-system-depending security system and the client request is executed under the client's account data.

Using the SECURITY_MODE parameter, you can specify at which rank (Batch) you want to impersonate the activities of an NDV client.

| Value | Explanation |
|---|---|
| IMPERSONATE_LOCAL | Impersonation is done within the Natural Development Server environment. If the session is dispatched in a remote TP environment, it is still executed anonymous. The client must be defined in the security system of the NDV server. It is not required to define the client in a remote TP environment. |
| IMPERSONATE_REMOTE | No impersonation is done within the Natural Development Server environment. If the session is dispatched in a remote TP environment, the client is impersonated. The client must be defined in the security system of the remote TP environment. |
| IMPERSONATE | Impersonation is done within the Natural Development Server environment and in a remote TP environment. The client must be defined in the security system of the NDV server and in the remote TP environment. |

**Note:** For a batch server SECURITY_MODE=IMPERSONATE and SECURITY_MODE=IMPERSONATE_LOCAL are the same.

No default value is provided.

Example:

```
SECURITY_MODE=IMPERSONATE
```

### SESSION_PARAMETER

This optional configuration parameter defines session parameters that precede the parameter string either specified in the **Map Environment** dialog of Natural Studio or defined by default by the configuration parameter DEFAULT_PROFILE.

| Value | Explanation |
|---|---|
| *parameter-string* | This string may extend across several lines. A plus sign (+) at the end of a string line denotes that another line follows. |

No default value is provided.

Example 1:

```
SESSION_PARAMETER='NUCNAME=NATNUCvr' +
  'PROFILE=(NDVPARM,18006,48),ADAMODE=0,' +
  'BPI=(TYPE=NAT,SIZE=6044),BPI=(TYPE=EDIT,SIZE=2048)', +
  'BPI=(TYPE=SORT,SIZE=1024)'
```

- where *vr* stands for the version and release number.

Example 2:

```
SESSION_PARAMETER=FNAT=(10,930)
```

The setting in the second example defines that every session on this Natural Development Server is started with the session parameter `FNAT=(10,930)` appended to the user-specified parameters or the definitions in the configuration parameter `DEFAULT_PROFILE`.

### SESSION_PARAMETER_MIXED_CASE

This optional configuration parameter can be used to allow session parameters and URL specifications in mixed case.

| Value | Explanation |
|---|---|
| `YES` | Session parameters remain in mixed case. |
| `NO` | Session parameters are translated into upper case. This is the default value. |

### SESSION_TIMEOUT

Cancel inactive sessions when the `SESSION_TIMEOUT` parameter is met. Check for sessions inactive longer then *n* minutes once a day at `HH:MM` (24 hours) or every *n* minutes.

The server will not start if an invalid `SESSION_TIMEOUT` parameter is given.

| Value | Explanation |
|---|---|
| `hh:mm,n <numeric value greater than 0>` or<br><br>`m <numeric value greater than 0>,n <numeric value>0>` | If format is `hh:mm`, check once a day at `hh:mm` for sessions more than *n* minutes inactive.<br><br>or<br><br>If format is a numeric value, check every *m* minutes for sessions more than *n* minutes inactive. |

Examples:

```
SESSION_TIMEOUT=19:30,480
```

Every day at 19:30 cancel sessions more than 480 minutes inactive.

```
SESSION_TIMEOUT=360,480
```

Every 360 minutes cancel sessions more than 480 minutes inactive.

### TERMINAL_EMULATION

This configuration parameter defines the terminal emulation to be used for processing the Natural I/O. This definition applies to all clients using that server.

| Value | Explanation |
|---|---|
| `WEBIO` | Use the Web I/O Interface as terminal emulation. |
| `3270` | Use the 3270 terminal emulation. This is the default value. |

Example:

```
TERMINAL_EMULATION=WEBIO
```

### TRACE_FILTER

This optional configuration parameter enables you to restrict the trace by a logical filter in order to reduce the volume of the server trace output, for example:

```
TRACE_FILTER="Client=(KSP P*)"
```

Each request of the user ID `KSP` and each request of the user IDs starting with a `P` are traced.

See ***Trace Filter*** in the section *Operating the Natural Development Server*.

### TRACE_LEVEL

| Value | Explanation |
|---|---|
| *`trace-level`* | See ***Trace Level*** in the section *Operating the Natural Development Server*. |
| `0` | This is the default value. |

Example:

```
TRACE_LEVEL=0x00000011
```

or alternatively

```
TRACE_LEVEL=31+27
```

The setting in the example switches on **Bits 31 and 27**.

### UNICODE_SOURCE

This configuration parameter is used to define whether the NDV server accepts source files in Unicode or not.

Sources transmitted in unicode are not converted using the Natural ASCII-to-EBCDIC translation tables `TABA1/TABA2`. All characters in the source file are supported without maintaining the Natural translation tables.

A transmission in Unicode, however, increases the CPU consumption of the server significantly.

| Value | Explanation |
|---|---|
| `YES` | Transfer sources in Unicode. |
| `NO` | Transfer sources in ASCII. No code page support for Natural sources. This is the default value. |

Example:

```
UNICODE_SOURCE=YES
```

### UPPERCASE_SYSTEMMESSAGES

This configuration parameter is used to enable or disable the translation of all NDV error messages and trace outputs to uppercase. This feature is for customers who are using character sets with no lowercase characters defined.

| Value | Explanation |
|---|---|
| `YES` | Enable uppercase translation. |
| `NO` | Disable uppercase translation. This is the default value. |

## NDV Configuration File Example

```
# This is a comment
SESSION_PARAMETER=profile=(stgqa,10,930) fuser=(10,32)
DEFAULT_PROFILE=DEFPROF
FRONTEND_NAME=NATNCF       # and another comment
PORT_NUMBER=4711
```

## NDV Server Datasets

The Natural Development Server requires the following datasets:

| | |
|---|---|
| STGCONFG | Defines the server configuration file. |
| STGTRACE | The server trace output. |
| STGSTDO | The stdo dataset. |
| STGSTDE | The stde error output. |

Alternatively, you can qualify each dataset name by the server ID. This is necessary if you want to start different Natural Development Servers under a single SMARTS address space.

| | |
|---|---|
| NDVS1C | Defines the server configuration file for the server NDVS1. |
| NDVS1T | The server trace output for the server NDVS1. |
| NDVS1O | The stdo dataset for the server NDVS1. |
| NDVS1E | The stde error output for the server NDVS1. |

## NDV User Exits (Coded in Natural)

Natural Single Point of Development provides the following user exits for mainframes:

| | |
|---|---|
| NDV-UX01 | This exit is invoked before a Natural source object or a DDM is edited. It can be used to reject editing of certain sources. The source code of this exit is delivered in the library SYSLIB and named NDV-SX01 *). |
| NDV-UX02 | This exit is invoked before a Natural object, a DDM or a user error message is deleted, copied or moved (including the context menu functions Cut, Copy and Paste). It enables the rejection of further processing of this object, similar to the user exit MAINEX01 of SYSMAIN in Natural for Mainframes. The source code of this exit is delivered in the library SYSLIB and named NDV-SX02 *). |

| | |
|---|---|
| NDV-UX03 | This exit provides flags for special settings within the Natural Development Server. See the source code of this exit for available flags. The source code of this exit is delivered in the libary SYSLIB and is named NDV-SX03 *). |

*) The sources of these user exit routines are named NDV-SX*nn*, where *nn* denotes the number of the user exit routine.

**To make a user exit routine available**

1 Copy the source code from SYSLIB into a user library.

2 Catalog it under the name NDV-UX*nn*.

3 Copy it back into the Natural system library SYSLIB.

The name of each user exit source is different from the name of the corresponding cataloged object. This guarantees that the object is not affected if the user exit source is overwritten by an installation update.

For further details, see the source code of the user exit routines NDV-SX*nn* in the Natural system library SYSLIB.

## Other NDV User Exits

Apart from the NDV user exits that are coded in Natural, the following user exit exists:

### User Exit NSECUX01

This user exit is applicable only when the parameter SECURITY_MODE is set to IMPERSONATE_LOCAL or IMPERSONATE.

This user exit allows you to adapt the user ID used for the RACF login. It is useful if the RACF user IDs and the user IDs used in Natural differ according to a standardized rule. For example, each RACF user ID is the corresponding Natural user ID preceded by two dollar signs ($$).

If the exit (the loadmodule NSECUX01) is found in the NDV load library concatenation, it is called using standard linkage conventions (direct branch using BASR instructions) before the user is validated against RACF.

The following parameters are passed to the exit:

| Name | Format | In/Out | Description |
|---|---|---|---|
| sUid | CL64 | I/O | User ID to be modified for RACF login. |

The exit is called using standard linkage conventions.

Sample user exit implemented in C:

```
#include <string.h>
#include <stdio.h>
#  pragma linkage (NSECUX01, FETCHABLE)
void NSECUX01(char sUid[64])
{
char sUidTemp[64];
  printf("Uex got usid:%s\n", sUid);
  strcpy(sUidTemp, sUid);
  sprintf(sUid, "$$%s", sUidTemp);
  printf("Uex ret usid:%s\n", sUid);
  return;
}
```

The exit above extends each user ID by two preceding dollar signs ($$) when it is used for RACF login.

# 8 Operating the Natural Development Server


- Starting the Natural Development Server ..... 46
- Terminating the Natural Development Server ..... 46
- Monitoring the Natural Development Server ..... 47
- Runtime Trace Facility ..... 48
- Trace Filter ..... 49

This chapter describes how to operate a Natural Development Server in a Com-plete on z/OS environment.

## Starting the Natural Development Server

A prerequisite is a running Com-plete address space that is configured to run the Natural Development Server (see *Development Server Installation*).

Start the Natural Development Server with the Com-plete console command

```
F name,NATRDEVS <server-id>
```

- where

*name* is the address space name assigned to the Com-plete region,

*server-id* is the name of your Natural Development Server.

**Example:**

```
F COMPLETE,NATRDEVS NDVS1
```

**Note:** If you qualify the Natural Development Server datasets by *server-id*, the server ID is restricted to a maximum length of 6 characters.

Alternatively, you can automatically start Natural Development Servers during Com-plete initialization by using the Com-plete SYSPARM parameter STARTUPPGM. In the Com-plete SYSPARM file specify:

```
STARTUPPGM='NATRDEVS <server-id>'
```

**Example:**

```
STARTUPPGM='NATRDEVS NDVS1'
```

## Terminating the Natural Development Server

The Natural Development Server can be terminated from within the **Monitor Client NATMOPI**, see ***Monitor Commands***.

# Monitoring the Natural Development Server

To enable the administrator to monitor the status of the Natural Development Server, a monitor task is provided which is initialized automatically at server startup. Using the monitor commands described below, the administrator can control the server activities, cancel particular user sessions, terminate the entire server, etc.


- Monitor Communication
- Monitor Commands


## Monitor Communication

To communicate with the monitor, you can use the monitor client `NATMOPI`; see ***Monitor Client NATMOPI***.

Alternatively, you can use the operator command `MODIFY` to execute the monitor commands described below in the section ***Monitor Commands***. The output of the executed monitor command will be written to the system log.

**Example:**

```
F jobname,APPL=ping
```

sends the command `ping` to the NDV server running under the job *jobname*.

## Monitor Commands

The Natural Development Server supports the following monitor commands:

| Monitor Command | Action |
|---|---|
| `ping` | Verifies whether the server is active. The server responds and sends the string<br><br>`I'm still up` |
| `terminate` | Terminates the server. |
| `abort` | Terminates the server immediately without releasing any resources. |

| Monitor Command | Action |
|---|---|
| `set configvariable value` | With the `set` command, you can modify server configuration settings. For example, to modify `TRACE_LEVEL`:<br><br>`set TRACE_LEVEL 31+30+15` |
| `list sessions` | Returns a list of active Natural sessions within the server. For each session, the server returns information about the user who owns the session, the session initialization time, the last activity time and an internal session identifier (*session-id*). |
| `cancel session session-id` | Cancels a specific Natural session within the Natural Development Server. To obtain the session ID, use the monitor command `list sessions`. |
| `cleanup` | Cancel sessions that are inactive longer than specified in configuration parameter `SESSION_TIMEOUT`. |
| `help` | Returns help information about the monitor commands supported. |

## Runtime Trace Facility

For debugging purposes, the server code has a built-in trace facility which can be switched on, if desired.

- Trace Medium
- Trace Configuration
- Trace Level

### Trace Medium

A remote development server writes its runtime trace to Com-plete job output into the dataset *srvname*`T`.

### Trace Configuration

The trace is configured by a **trace level** which defines the details of the trace. Once a trace is switched on, it can be restricted to particular clients or client requests by specifying a **trace filter**, see also NDV configuration parameter `TRACE_FILTER`.

Every Natural session is provided with a 32-bit trace status word (TSW) which defines the trace level for this session. The value of the TSW is set in the NDV configuration parameter `TRACE_LEVEL`. A value of zero means that the trace is switched off.

### Trace Level

Each bit of the TSW is responsible for certain trace information. Starting with the rightmost bit:

| | |
|---|---|
| Bit 31 | Trace main events (server initialization/termination, client request/result). |
| Bit 30 | Detailed functions (session allocation, rollin/rollout calls, detailed request processing). |
| Bit 29 | Dump internal storage areas. |
| Bit 28 | Session directory access. |
| Bit 27 | Dump request/reply buffer EBCDIC. |
| Bit 26 | Dump request/reply buffer ASCII. |
| Bit 25 | Dump I/O buffer. |
| Bit 24 | Free. |
| Bit 23 | Request processing main events. |
| Bit 22 | Request processing detailed functions. |
| Bit 21 | Remote debugger main events. |
| Bit 20 | Remote debugger detailed functions. |
| Bit 19-16 | Free. |
| Bit 15 | Trace error situations only. |
| Bit 14 | Apply trace filter definitions. |
| Bit 13 | Trace start and termination of the server only. |
| Bit 12 | Trace start and termination of the client sessions only. Even if bit 13 is set. |
| Bit 11-08 | Free. |
| Bit 07-01 | Free. |
| Bit 00 | Reserved for trace-level extension. |

## Trace Filter

It is possible to restrict the trace by a logical filter in order to reduce the volume of the server trace output.

- The filter can be set with the configuration parameter `TRACE_FILTER`.
- The filter may consist of multiple *keyword=filtervalue* assignments separated by spaces.
- To activate the filter definition, the trace bit 14 in the trace status word (see ***Trace Level***) must be set.

The filter keyword is:

| | |
|---|---|
| `Client` | Filters the trace output by specific clients. |

The following rules apply:

- If a keyword is defined multiple times, the values are cumulated.
- The value must be enclosed in braces and can be a list of filter values separated by spaces.
- The values are not case sensitive.
- Asterisk notation is possible.

**Example:**

```
TRACE_FILTER="Client=(KSP P*)"
```

Each request of the user ID `KSP` and each request of the user IDs starting with a `P` are traced.

# 9 Monitor Client NATMOPI


- Introduction ..... 52
- Command Interface Syntax ..... 52
- Command Options Available ..... 52
- Monitor Commands ..... 53
- Directory Commands ..... 53
- Command Examples ..... 54

## Introduction

The Monitor Client NATMOPI is a character-based command interface for monitoring the various types of servers that are provided in a mainframe Natural environment. Each of these servers has its own set of monitor commands which is described in the corresponding server documentation. In addition, a set of directory commands is available which can be used independent of the server type. One NATMOPI can be used to monitor different server types.

## Command Interface Syntax

Basically the syntax of the command interface consists of a list of options where each option can/must have a value. For example:

```
-s <server-id> -c help
```

where -s and -c are options and <server-id> and help are the option values.

It is possible to specify multiple options, but each option can have only one value assigned.

The command options available are listed below.

## Command Options Available

Words enclosed in <> are user supplied values.

| Command Option | Action |
|---|---|
| -s <server-id> | Specify a server ID for sending a **monitor command**. If the server ID is not unique in the server directory, NATMOPI prompts the user to select a server. |
| -c <monitor command> | Specify a **monitor command** to be sent to the server ID defined with the -s option |
| -d <directory command> | Specify a **directory command** to be executed. |
| -a | Suppress prompting for ambiguous server ID. Process all servers which apply to the specified server ID. |
| -h | Print NATMOPI help. |

## Monitor Commands

These are commands that are sent to a server for execution. The monitor commands available depend on the type of server, however, each server is able to support at least the commands `ping`, `terminate` and `help`.

For further commands, refer to *Operating the Natural Development Server* where the corresponding **server commands** are described.

## Directory Commands

Directory commands are not executed by a server, but directly by the monitor client `NATMOPI`.

You can use the directory commands to browse through the existing server entries and to remove stuck entries.

The following directory commands are available. Words enclosed in <> are user supplied values and words enclosed in [] are optional.

| Directory Command | Action |
|---|---|
| `ls [<server-id>]` | List all servers from the server directory that apply to the specified server ID. The server list is in short form. |
| `ll [<server-id>]` | Same as `ls`, but the server list contains extended server information. |
| `rs [<server-id>]` | Remove server entries from server directory.<br>**Note:** If you remove the entry of an active server, you will loose the ability to monitor this server process. |
| `cl [<server-id>]` | Clean up server directory. This command pings the specified server. If the server does not respond, its entry will be removed from the directory. |
| `ds` | Dump the content of the server directory. |
| `lm` | List pending IPC messages. |

# Command Examples

### Example: Ping a Server in Different Environments

Server in z/OS (SMARTS/Com-plete), output in `STDOUT`:

- System operator:

```
/F complete-jobname,NATMOPI -sServerName -cPING
```

- Com-plete online command:

```
NATMOPI -sServerName -cPING
```

### Further Command Examples:

| | |
|---|---|
| `natmopi -dls` | List all servers registered in the directory in short format. |
| `natmopi -dcl TST -ls TST` | Clean up all servers with ID `TST*` (ping server and remove it, if it does not respond), and list all servers with ID `TST*` after cleanup. |
| `natmopi -sSRV1 -cping -sSRV2 ↩ -sSRV3 -cterminate` | Send command `ping` to `SRV1`. Send command `terminate` to `SRV2` and `SRV3`. |
| `natmopi -cterminate -sSRV1 ↩ -cping -sSRV2 -sSRV3` | Is equivalent to the previous example. That is, `NATMOPI` sends the command following the `-s` option to the server. If no `-c` option follows the `-s` option, the first `-c` option from the command line will be used. |
| `natmopi -sSRV1 -cterminate -a` | Send command `terminate` to `SRV1`. If `SRV1` is ambiguous in the server directory, send the command to all `SRV1` servers without prompting for selection. |

# 10 HTML Monitor Client


- Introduction ..... 56
- Prerequisites for HTML Monitor Client ..... 56
- Server List ..... 56
- Server Monitor ..... 58

## Introduction

The HTML Monitor Client is a monitor interface that supports any web browser as a user interface for monitoring the various types of servers that are provided in a mainframe Natural environment. Each of these servers has its own set of monitor details which are described in the corresponding server documentation. The HTML Monitor Client enables you to list all existing servers and to select a server for monitoring.

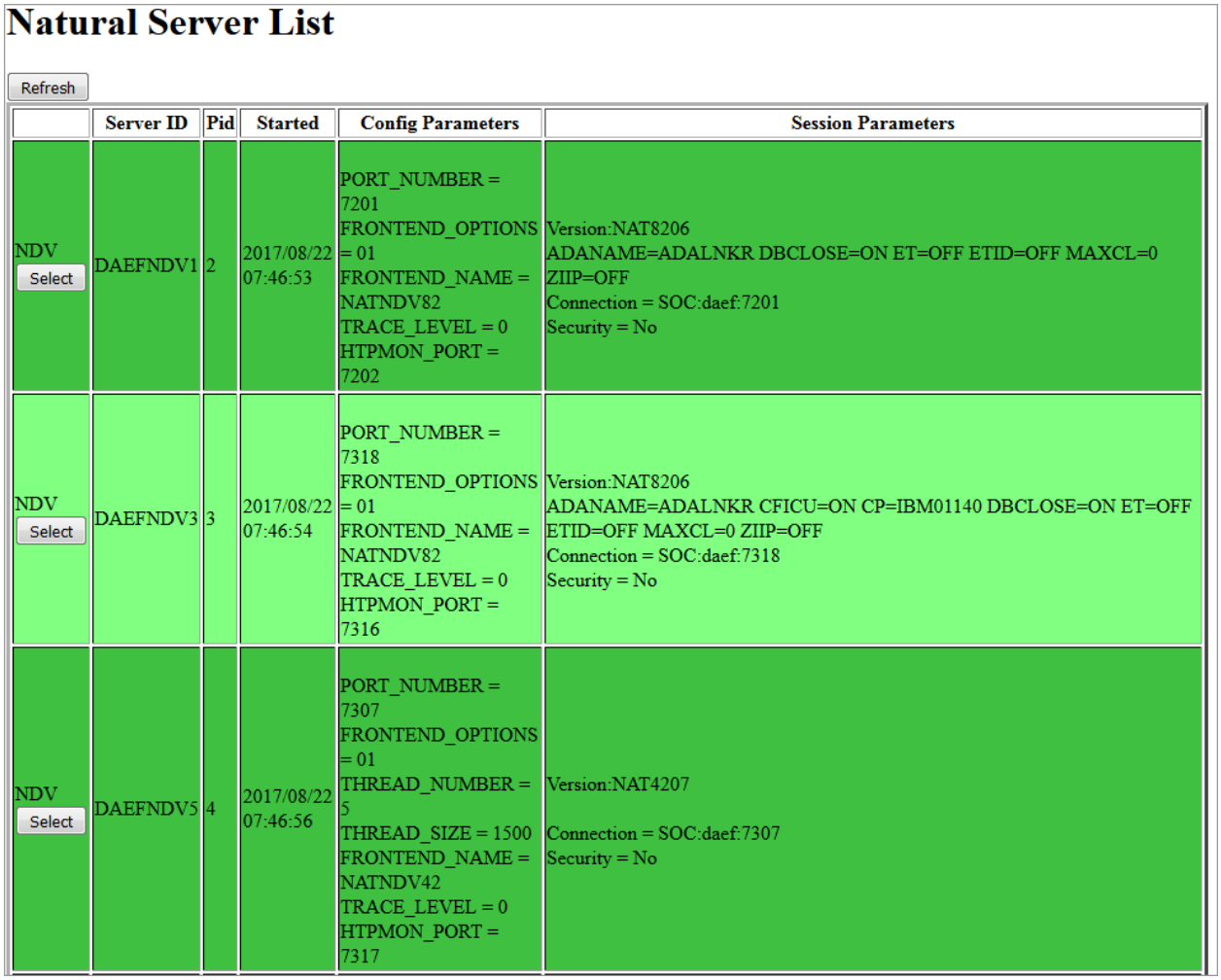
## Prerequisites for HTML Monitor Client

To run the HTML Monitor Client, any server must host an HTTP Monitor Server. The HTTP Monitor Server is a subtask that can run in any Natural Development Server address space and is configured with the NDV configuration parameter `HTPMON_PORT` and `HTPMON_ADMIN_PSW`. An HTTP Monitor Server is accessible through a TCP/IP port number and can monitor all servers running on the current node (for SMARTS: running within the current SMARTS). Although it is not necessary, you can run multiple HTTP Monitor Servers on one node. But each one needs an exclusive port number.

## Server List

Open your web browser and connect the HTTP Monitor Server using the following URL: `http://`*`nodename`*`:`*`port`*, where *`nodename`* is the name of the host on which the Natural Development Server hosting the monitor is running. And *`port`* is the port number the administrator has assigned as the monitor port in the NDV configuration file.

**Example**



Natural Server List

Refresh

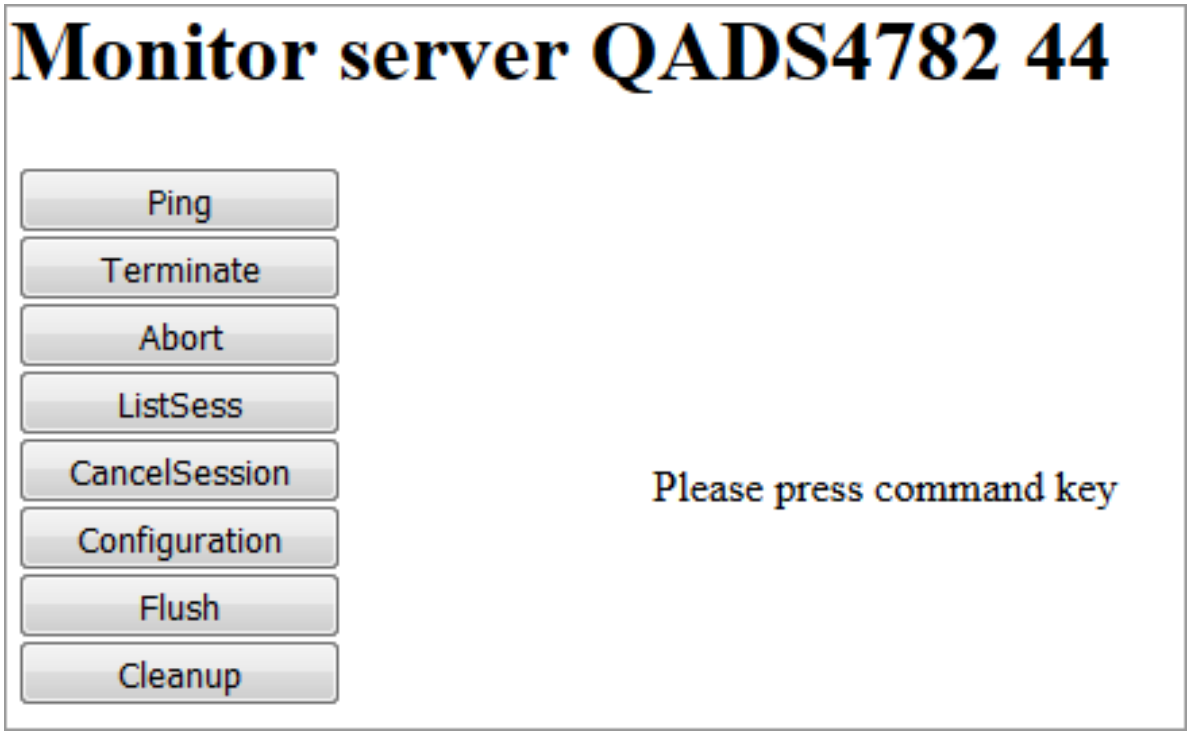
| | Server ID | Pid | Started | Config Parameters | Session Parameters |
|---|---|---|---|---|---|
| NDV Select | DAEFNDV1 | 2 | 2017/08/22 07:46:53 | PORT_NUMBER = 7201 FRONTEND_OPTIONS = 01 FRONTEND_NAME = NATNDV82 TRACE_LEVEL = 0 HTPMON_PORT = 7202 | Version:NAT8206 ADANAME=ADALNKR DBCLOSE=ON ET=OFF ETID=OFF MAXCL=0 ZIIP=OFF Connection = SOC:daef:7201 Security = No |
| NDV Select | DAEFNDV3 | 3 | 2017/08/22 07:46:54 | PORT_NUMBER = 7318 FRONTEND_OPTIONS = 01 FRONTEND_NAME = NATNDV82 TRACE_LEVEL = 0 HTPMON_PORT = 7316 | Version:NAT8206 ADANAME=ADALNKR CFICU=ON CP=IBM01140 DBCLOSE=ON ET=OFF ETID=OFF MAXCL=0 ZIIP=OFF Connection = SOC:daef:7318 Security = No |
| NDV Select | DAEFNDV5 | 4 | 2017/08/22 07:46:56 | PORT_NUMBER = 7307 FRONTEND_OPTIONS = 01 THREAD_NUMBER = 5 THREAD_SIZE = 1500 FRONTEND_NAME = NATNDV42 TRACE_LEVEL = 0 HTPMON_PORT = 7317 | Version:NAT4207 Connection = SOC:daef:7307 Security = No |

The server list consists of green and red entries. The red ones represent potentially dead server entries which can be deleted from the server directory by choosing the attached **Remove** button. The **Remove** button appears only for the red entries. “Potentially dead” means, that the HTTP Monitor Server “pinged” the server while assembling the server list, but the server did not answer within a 10 seconds timeout. Thus, even if you find a server entry marked red, it still might be active but could not respond to the ping. Choosing the **Remove** button does not terminate such a server but removes its reference in the monitor directory. Hence, it cannot be reached by the monitor anymore.

Choosing the **Select** button opens a window for monitoring the selected server.

## Server Monitor

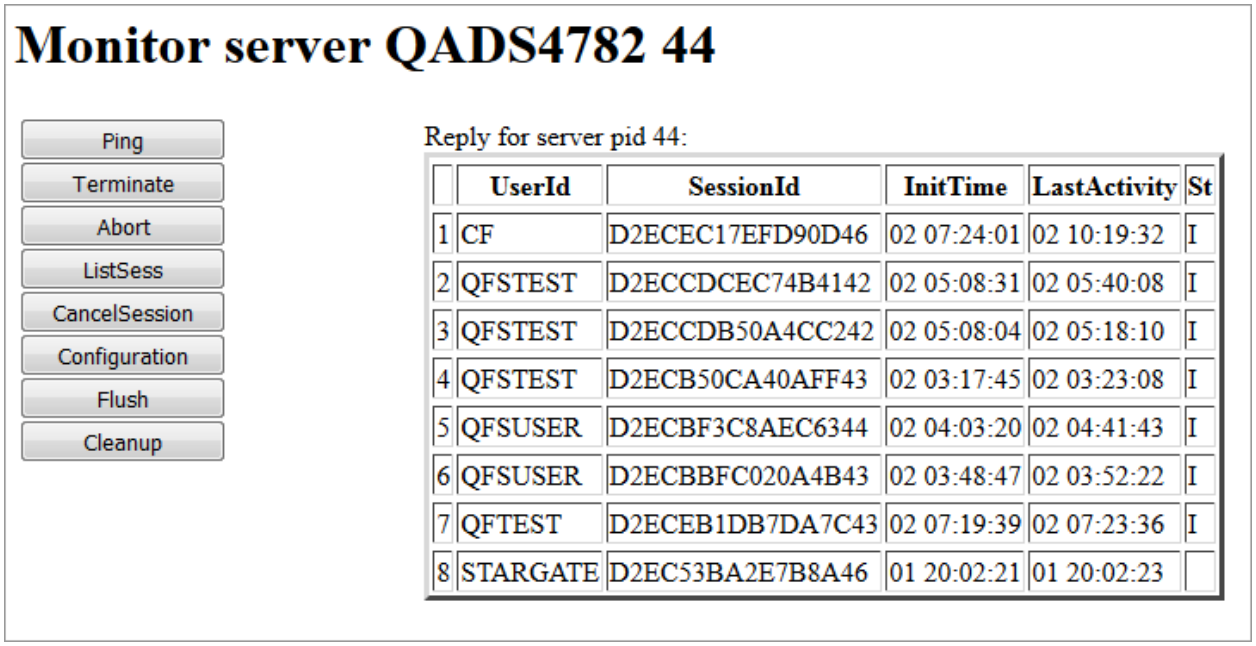
Example:




With the buttons, you can perform the labeled monitor commands.

The selection box allows you to modify the server configuration parameters. If you select a parameter for modification, it has a predefined value. This predefined value does not reflect the setting of the server. It is just a sample value.

If you choose the **ListSess** button, a list of all Natural sessions appears in the window, for example:




| | UserId | SessionId | InitTime | LastActivity | St |
|---|---|---|---|---|---|
| 1 | CF | D2ECEC17EFD90D46 | 02 07:24:01 | 02 10:19:32 | I |
| 2 | QFSTEST | D2ECCDCEC74B4142 | 02 05:08:31 | 02 05:40:08 | I |
| 3 | QFSTEST | D2ECCDB50A4CC242 | 02 05:08:04 | 02 05:18:10 | I |
| 4 | QFSTEST | D2ECB50CA40AFF43 | 02 03:17:45 | 02 03:23:08 | I |
| 5 | QFSUSER | D2ECBF3C8AEC6344 | 02 04:03:20 | 02 04:41:43 | I |
| 6 | QFSUSER | D2ECBBFC020A4B43 | 02 03:48:47 | 02 03:52:22 | I |
| 7 | QFTEST | D2ECEB1DB7DA7C43 | 02 07:19:39 | 02 07:23:36 | I |
| 8 | STARGATE | D2EC53BA2E7B8A46 | 01 20:02:21 | 01 20:02:23 | |

You can cancel sessions by selecting the session ID in the **SessionId** column and choosing the **CancelSession** button.

# 11 SPoD-Specific Limitations and Considerations


- Limitations ..... 62
- Performance Considerations ..... 71
- Natural Documentation and Online Help ..... 75

When you are working with Natural Single Point of Development, you will encounter a few limitations which are due to the different capabilities of the graphical user interface available on the local site and the character-based user interface that exists on the remote site. In addition, this document includes hints which are important for the efficient use of the remote development facilities.

**Editor Features With SPoD**

You can use Natural's Single Point of Development with different versions of Natural on a variety of platforms. Depending on the server environment you are using together with Natural for Windows (client), the editors offer different features. For further information, refer to the section *Editor Features With SPoD* in the Natural for Windows *Editors* documentation.

# Limitations


- Back-End Program
- System Commands
- Profile Parameters
- Terminal Commands
- Moving/Copying Error Messages
- LIST DDM, EDIT DDM
- Maps Containing GUI Elements
- Field Sensitive Maps
- Resources
- Dialogs
- Natural ISPF Macros and Recordings
- SYSLIB/SYSLIBS
- Allow Lower Case Input in Program Editor of Natural Studio
- Terminal Emulation
- Dependencies between XRef Evaluation and Predict
- Remote Debugging


## Back-End Program

If a back-end program has been specified (for example, by means of the Natural profile parameter `PROGRAM`), it is not invoked if the Natural session is executing on a Natural Development Server.

## System Commands


- System Command SYSDDM
- System Commands Unavailable for Remote Development
- System Commands Entered Directly on the Development Server


### System Command SYSDDM

The system command `SYSDDM` is not available, since the DDMs are listed in the tree view under the node DDM, and because all functions of the utility `SYSDDM` are available by using Natural Studio's context menu or menu bar.

### System Commands Unavailable for Remote Development

The following system commands are not available, since their use would make no sense with a graphical user interface:

- `EDT`
- `HELLO`
- `MAINMENU`

### System Commands Entered Directly on the Development Server

All system commands which are not entered in the user interface of Natural Studio are executed directly by the Development Server without control of Natural Studio. As a result, the character-based representation of the corresponding command appears in the terminal emulation window. This is the case when the `STACK TOP COMMAND` mechanism is used or when a system command is directly entered inside the terminal emulation window.

During the mapping phase any `STACK` commands entered in the text box **Session Parameters** are processed within Natural Studio and the corresponding Natural Studio windows are used.

It is even possible to invoke the mainframe editors. However, this may lead to inconsistencies (see also *Object Locking* in the Natural for Windows documentation). Therefore, you are strongly recommended to use only Natural Studio's GUI editors.

The commands `HELLO` and `MAINMENU` do not cause a screen output on the development server side, since this would not make any sense in the SPoD environment. Instead of the menu-driven user interface, the dialogs provided in Natural Studio are used.

## Profile Parameters

### CP Parameter

The Natural profile parameter setting `CP=AUTO` is not supported in a SPoD environment. (`AUTO` means that the code page name from the user terminal is taken, if available.)

## Terminal Commands

Using terminal commands in a SPoD environment is only possible within the terminal emulation window. Entering terminal commands in the command line of Natural Studio is not possible.

## Moving/Copying Error Messages

Moving and copying of error messages is different in remote and local environments:

- When error messages are moved or copied within the remote environment or are moved or copied from the local to the remote environment or vice versa: the error messages involved are merged, that is,
  - error messages which already exist in the target environment are replaced,
  - messages which do not exist in the source library are kept in the target library,
  - messages which do not exist in the target library are added.
- When error messages are moved or copied within the local environment, the messages involved are handled on file level, that is,
  - all error messages (that is, files) of a language are deleted and
  - the file from the source library is created anew in the target library.

## LIST DDM, EDIT DDM

In contrast with a pure Natural mainframe environment, that is, without remote development from Natural Studio, the command `EDIT DDM` is available also from a user library. This means that it is not necessary to expand the DDM node in the tree view to be able to edit a specific DDM. However, when Natural Security is used, the use of the commands `LIST DDM` and `EDIT DDM` can be restricted only via the security profile of the mainframe Natural utility `SYSDDM`.

### Maps Containing GUI Elements

Maps containing GUI elements can be moved or copied from the local environment to a remote environment. However, the GUI elements are not displayed when the map is being tested or executed on the remote environment.

### Field Sensitive Maps

For these maps, the consistency check for a map field is made as soon as the user input has been entered. Field sensitive maps can be moved or copied from the local environment to a remote environment. However, a field sensitive map cannot be tested or executed on a remote mainframe environment.

### Resources

On the mainframe, objects of type resource can be handled (displayed, copied, deleted, etc.). See *Using Non-Natural Files - Resources* in the Natural *Programming Guide*.

By default, resouces are not handled by Natural Development Server for performance reasons; that is, resources are normally *not* displayed.

If you want Natural Development Server to handle resources, use the user exit `NDV-UX03` (source: `NDV-SX03`), which allows you to enable/disable the display of resources in Natural Studio for all or certain users only.

The server behaves in the following way: If the user exit exists and the flag `DISPLAY-RESOURCES` contained therein is set (`Y`), the server checks in the Library Statistical Record whether the number of resources is greater than 0. If so, the library is searched also for resources.

### Dialogs

Dialogs can be stored on the mainframe. Therefore it is possible to move or copy dialogs from the local environment to a remote environment. Private resource files of a dialog will not be moved or copied together with the dialog. It is also possible to list dialogs in a remote environment. New dialogs cannot be created and dialogs cannot be edited in a remote environment.

### Natural ISPF Macros and Recordings

As the object types Natural ISPF Macro and Recording available with Natural for Mainframes cannot be processed by Natural Studio, they will not be displayed in the tree view of the library work space. If a library consists only of such object types, the library will be displayed nevertheless in the tree view, but without any subnodes.

If a library containing such object types is deleted, then the objects of these two specific object types will not be deleted and the library will continue to be displayed in the tree view.

Objects of the types Natural ISPF Macro and Natural ISPF Recording cannot be linked to an application.

### SYSLIB/SYSLIBS

The restricted libraries `SYSLIB`/`SYSLIBS` of the server are not shown in Natural Studio's tree view, because a logon to these libraries is not possible. These libraries can be modified only by using a Natural utility such as `SYSMAIN` or the Object Handler.

### Allow Lower Case Input in Program Editor of Natural Studio

Natural Studio's program editor is case-sensitive, that is, lower case input will be included in the program source in lower case. The compiler on the Natural Development Server, however, expects upper case code in its normal setting. This issue can be fixed by setting the compiler option `LOWSRCE=ON`. But this setting will have specific side effects which should be noticed. Refer to the `CMPO` profile parameter in the Natural *Parameter Reference*.

### Terminal Emulation

The terminal emulation supports 3270 Model 2 screens. The support of 3270 Model 3, 4 and 5 screens is planned for one of the next versions of Natural Single Point of Development.

### Dependencies between XRef Evaluation and Predict

If you are using dynamic language assigned when calling other objects such as `INPUT USING MAP 'MAP1&'`, the connection between caller and called object cannot be retrieved by using XRef Evaluation.

Natural on the mainframe supports case-sensitive calls to other objects such as `PERFORM SUBROUTINE`. With the current version of SPoD, this may lead to strange results when, in XRef Evaluation, trees are expanded and it is not possible to request case-sensitive calls with the filter dialog.

## Remote Debugging

When the remote debugging facility was implemented, the goal was not to provide any new functions, but to support the existing essential debugging functions under the Natural Development Server. These functions are:

- Stepmode
- Breakpoints
- Watchpoints
- Display and modification of variables and their contents during a break.

Generally, it was intended to provide for compatibility between the debug functionality that exists in a Natural on mainframes and a Natural on PC. Hence, the current state of development constitutes the lowest possible common denominator. Especially the debug statistics as supported on mainframes are not yet supported in a remote debug environment.

### Which Differences Exist in Debugging in a Mainframe Environment and in Natural Studio?

The following tables provide an overview of differences that exist between Natural debugging in a mainframe environment and debugging in Natural Studio.

Explanation of the table headings:

| | |
|---|---|
| **MF** | Describes debugging functionality available or not available in a mainframe Natural environment. |
| **SPoD** | Describes debugging functionality available or not available in a Natural Single Point of Development environment using Natural Studio as a development client and a mainframe Natural Development Server. |
| **PC** | Describes debugging functionality available or not available in Natural Studio (stand alone). |

- **Restarting a Debug Process**

| | |
|---|---|
| **MF** | The restart function is not supported. |
| **SPoD** | The restart function is not supported. |
| **PC** | Debug on PC offers a special restart function which is not available for remote debugging on mainframe. |

- **EXIT from Debugger**

| | |
|---|---|
| **MF** | System command `RUN`: leave the Natural Debugger. The program execution continues. |
| | System command `STOP`: both debugging and execution are terminated. |
| **SPoD** | Stop command in Debug menu: debug mode terminates, program execution stops. |
| **PC** | Stop command in Debug menu: debug mode terminates, program execution stops. |

### STEP OVER

| | |
|---|---|
| **MF** | Syntax of `STEP SKIPSUBLEVEL` is used instead of `STEP OVER`. |
| **SPoD** | `STEP OVER` is applicable for called objects on a different level (`CALLNAT`, etc). It is not applicable for internal subroutines. |
| **PC** | `STEP OVER` is supported for any called objects and, in addition, for internal subroutines. |

### Set Next Statement (Natural Studio Context Menu Command)

| | |
|---|---|
| **MF** | Not applicable. |
| **SPoD** | Not supported. |
| **PC** | Supported. Allows you to continue the execution at a chosen line. |

### System Variables: Display/Modify

| | |
|---|---|
| **MF** | System variables can be displayed, but cannot be modified. |
| **SPoD** | System variables can be displayed, but cannot be modified. |
| **PC** | System variables can be displayed and modified. |

### Display of Binary Variables

| | |
|---|---|
| **MF** | Either alphanumeric or hexadecimal display of binary variables can be selected. In alphanumeric display, binary variables with lengths ranging from 1 to 4 are interpreted and displayed as numerical values. Binary variables with lengths > 4 are displayed in alphanumeric representation. |
| **SPoD** | Binary variables are always represented as hexadecimal values. |
| **PC** | Binary variables are always represented as hexadecimal values. |

### Modify Dynamic Variables

| | |
|---|---|
| **MF** | During the debug process, the content of a dynamic variable can be modified in the given length. Modification of length of dynamic variable during debug is not supported. |
| **SPoD** | Content of dynamic variable can be modified in given length. |
| **PC** | Both content and length of dynamic variable can be modified during the debug process. |

### Maximum Length when Displaying Variable Values

| | |
|---|---|
| **MF** | Displays full content of variable; long variables are displayed in chunks of maximally 256 bytes. If Unicode is used: max. 256 bytes = 128 characters. |
| **SPoD** | Displays maximally 253 bytes. If Unicode is used: max. 252 bytes = 126 characters. |
| **PC** | Displays maximally 253 characters. |

### Maximum Length of Watchpoint Variables

| | |
|---|---|
| **MF** | Maximally 253 bytes. If Unicode is used: max. 252 bytes = 126 characters. |
| **SPoD** | Maximally 253 bytes. If Unicode is used: max. 252 bytes = 126 characters. |
| **PC** | Maximally 253 characters. |

### Watchpoint: Display of Break Line

| | |
|---|---|
| **MF** | After the watchpoint has been registered, the Debugger marks the preceding (already executed) statement. |
| **SPoD** | After the watchpoint has been registered, the Debugger stops at the current position in the program. This is the statement to be executed next. |
| **PC** | After the watchpoint has been registered, the Debugger stops at the current position in the program. This is the statement to be executed next. |

### Several Watchpoint Breaks per Line of Program

| | |
|---|---|
| **MF** | Multiple breaks on the same line may arise for the same watchpoint variable (because of different watchpoint operators). The hit counter is incremented accordingly. |
| **SPoD** | Multiple breaks on the same line may arise for the same watchpoint variable (because of different watchpoint operators). The hit counter is incremented accordingly. |
| **PC** | Several watchpoint definitions for the same variable result in a maximum of one break per line, hit counts of all watchpoint definitions are incremented. |

### Breakpoint Definition

| | |
|---|---|
| **MF** | Breakpoints can only be defined for programs which are found in the current library or in any steplib. |
| **SPoD** | Breakpoints can only be defined for programs which are found in the current library or in any steplib. |
| **PC** | Breakpoints for programs can be defined in any library (not necessarily in the current library or steplib). |

### Breakpoints BEG and END

| | |
|---|---|
| **MF** | The symbolic breakpoints `BEG` and `END` (first and last executed statement) are supported. |
| **SPoD** | Breakpoints `BP-BEG` and `BP-END` are not supported. |
| **PC** | Breakpoints `BP-BEG` and `BP-END` are not supported. |

### Debugging of Programs which are Called through the Stack

| | |
|---|---|
| **MF** | Stacked programs can be debugged when any breakpoint or watchpoint has been defined, but they cannot be entered automatically in step mode. |
| **SPoD** | Programs on stack can be entered in step mode. |
| **PC** | Programs on stack can be entered in step mode. |

### Edit/Stow a Program during Debug

| | |
|---|---|
| **MF** | A NAT0932 (program version) error appears if during the debug process the debugged program was stowed and loaded into the buffer pool. |
| **SPoD** | A NAT0932 (program version) error appears if during the debug process the debugged program was stowed and loaded into the buffer pool. |
| **PC** | Change and stow of program during debug is possible. |

### Call Stack

| | |
|---|---|
| **MF** | The debug command `OBJCHAIN` displays a list of active programs and their levels. |
| **SPoD** | The current program and its level are displayed in the call stack window. |
| **PC** | All active programs and their levels are displayed in the call stack window. |

## Performance Considerations

The working situation displayed in the library workspace of Natural Studio is based on the representation of the entire user system files.

The tree view window opens when the user connects to the Natural Development Server. For this, the entire system file has to be analyzed and the corresponding information has to be transferred from the Natural Development Server to the Natural Studio client. In the case of very large system files, the build-up of the tree view window can be very time consuming. Status information displayed in the status bar keeps the user informed about the progress of the screen build-up operation. This is to avoid the impression that the connection to the Natural Development Server might be interrupted.

**Tip:** Switch on the status bar using the **View > Status Bar** function of the menu bar. Make sure that the transfer rate of your network is 100 Mbit/s at minimum.

Another possibility to reduce the amount of data read while mapping the environment is to supply filter definitions on system file or library level.

**Tip:** In the context menu of a system file and library node it is possible to apply filter definitions. Using these definitions on the client side, you can limit the number of libraries/objects displayed in the tree view.

In the default configuration of Natural Studio, all operations which result in a modification of the system file, for example, moving or copying objects, but also a `SAVE` or `STOW` command, will cause the tree view window contents to be refreshed, which can be a very time consuming process in the case of very large system files.

**Tip:** By default, the **Refresh** function is set to **Full automatic refresh**. Change the automatic refresh function by choosing **Optimized automatic refresh** or **No automatic refresh** in the context menu.

Since the tree view of the application workspace displays only the objects that are linked to the application, the build-up of its tree view screen is consequently considerably faster, which is another advantage of using the application workspace.

## Library Statistical Record

In a Natural Single Point of Development environment, either local Natural libraries are accessed or Natural Studio requests the library statistical data from the remote development server. In the local environment, the data are stored persistently in the FILEDIR structure of the library. In the case of a mainframe development server, Natural objects are stored in system files in the database and the requested statistical data of a library are not stored permanently.

In order to reduce the number of Adabas calls and to improve the performance, a statistical record has been introduced.

The program `NDVCSTAR` is provided to initialize a complete system file, a range of libraries or a single library on a system file with the **Library Statistical Record**s, see ***Initialization of Library Statistical Records***.


- Concept
- Data Consistency
- Restrictions
- Initialization of Library Statistical Records


### Concept

In a Natural Single Point of Development environment, a library statistical record is created and maintained for every library of the `FUSER` or `FNAT` system file. This statistical record resides on the same system file where the library resides and contains the following information for every library:

- Total number of objects
- Total number of all sources
- Total number of all cataloged objects
- Total number of objects for every object type
- Accumulated size of all sources
- Accumulated size of all cataloged objects
- Accumulated size of sources for every object type
- Accumulated size of all cataloged objects for every object type

Supported object types:

- Program
- Map
- GDA
- LDA
- PDA

- Subroutine
- Helproutine
- Subprogram
- Copycode (source only)
- Text (source only)
- Command Processor
- Dialog (source only)
- Class
- Error Message (source only)
- Function
- Adapter
- Resource

When Natural Studio requests the statistics for a library the first time, the library statistical record is created and saved in the appropriate system file. Once the library statistical record has been built, all requests from Natural Studio will be satisfied by reading and sending the contents of the statistical record instead of rebuilding the complete library statistics.

When the user initiates an explicit refresh for a library, the statistical record is rebuilt completely.

#### Data Consistency

The library statistical record of a mainframe development server is supported only in a Single Point of Development environment. The statistical record is always up to date if all system file modifications are initiated in this environment.

All commands or operations triggered by Natural Studio which will modify the system files (add new object or copy, move, delete, rename object, etc.) will update the library statistical record on the development server.

In addition, the library statistical record is regenerated if an object list for the whole library is requested or the statistical record for a given object type is updated if an object list for this type is requested.

To ensure consistency of the data in the library statistical records of your `FNAT` and `FUSER` system files, you are strongly recommended to make changes on the same `FNAT` and `FUSER` system file used in a Single Point of Development environment exclusively in that environment.

**Caution:** When working with Natural Studio, care must be taken to start all commands or utilities from within Natural Studio. It is not admissible to issue system commands in the terminal emulation window, for example, at a `MORE` prompt or in a command line. In such

a case, the library statistical data might become inconsistent. The same is true if you start a server application that directly changes the FNAT or FUSER system file.

Such inconsistencies may be resolved after the next regeneration (implicit rebuild via get object list or explicit refresh) of the library statistical record is forced.

#### Restrictions

Statistical records cannot be used for read-only system files. In this case, the old behavior is used.

#### Initialization of Library Statistical Records

In order to initialize the Library Statistical Records of a a complete system file, a range of libraries or a single library on a system file you can invoke the program NDVCSTAR.

The following options are provided:

| Option | Meaning | | Default value |
|---|---|---|---|
| Library name range | Possible values: | | * |
| | blank or * (asterisk) | All libraries. | |
| | *value*> | All libraries with names greater than or equal to *value*. | |
| | *value*< | All libraries with names less than or equal to *value*. | |
| DBID, FNR | The database ID (DBID) and file number (FNR) of the system file where the Natural libraries are stored. If no values (or 0) are specified, the current FUSER or FNAT system file is used. | | 0,0 |
| Password, Cipher | The password and cipher code of the Adabas file where the Natural libraries are stored. | | None |
| Update existing records | Specifies whether existing Library Statistical Records are to be processed. Possible values: | | N |
| | Y (yes) | Regenerate existing Library Statistical Records. | |
| | N (no) | Skip existing Library Statistical Records. | |
| Display library names | Specifies whether a processing message of the library currently processed is to be displayed. Possible values: | | 1 |
| | Y (yes) | Display processing messages. | |
| | N (no) | Display only error messages. | |

**To invoke the program NDVCSTAR**

- At a NEXT/MORE prompt or in a Natural command line, enter NDVCSTAR and press ENTER.

**To execute the program NDVCSTAR in batch mode**

- Enter `NDVCSTAR` followed by the desired options.

  Examples:

  ```
  NDVCSTAR *,1,47,,,N,Y
  ```

  On the system file with `DB=D=1`, `FNR=47`, this command creates, for all libraries that do not yet have a Library Statistical Record, a new one. Any existing library statistical records are skipped. For every library found, a processing message is displayed.

  ```
  NDVCSTAR ABC*,,,,,Y,Y
  ```

  This command creates or regenerates the library statistical record on the current `FUSER` system file for all libraries that start with `ABC`. For every library found, a processing message is displayed.

## Natural Documentation and Online Help

The following restrictions apply to the Natural documentation and the Windows-based online help when you are using a Natural Development Server (NDV) for remote development:

- The online help currently available with Natural Studio contains only the Natural for Windows documentation and the SPoD client documentation.
- Therefore this online help may describe Natural features which are not or not yet supported on the mainframe platform.
- Natural features that are available only on mainframes are missing.
- Particularly in the sections dealing with the Natural programming language, minor but important differences due to hardware platforms, operating systems, TP monitors, etc. may exist.

We ask you to refer to the Natural for Mainframes documentation set for full details.

# 12 Natural Development Server Frequently Asked Questions


- Natural Development Server starts and terminates immediately ..... 78
- Which dataset should I analyze to get error information? ..... 78
- Trace output shows: Cannot load Natural front-end ... ..... 79
- Trace output shows: Transport initialization failed, EDC8115I address already in use ..... 79
- Trace output shows: Error at: Template runtime connect ..... 79
- Definitions required in Natural Security ..... 80
- Map Environment fails with a NAT3048 ..... 80
- Map Environment fails with Stub RCnn ..... 80
- Special characters are not translated correctly ..... 82
- Characters are not displayed correctly in the terminal emulation of Natural Studio ..... 84
- How do I find out which hexadecimal value must be specified for TABA1/TABA2? ..... 84
- The modifications of TABA1/TABA2 do not apply to sources listed in the remote debugger ..... 85
- Accessing work files ..... 85
- Are there any Natural profile parameter settings required for NDV? ..... 85
- NAT9915 GETMAIN for thread storage failed ..... 85
- The NDV server consumes a lot of CPU time even if only a few clients are using it ..... 86
- I get a NAT0873 internal error at user authentication for Map Environment ..... 86
- The server fails to start with return code 4 and in the error log I find ‘Transport initialization failed‘ ..... 86

This chapter contains frequently asked questions concerning the Natural Development Server (NDV) under z/OS (Com-plete).

## Natural Development Server starts and terminates immediately

At server initialization, the Natural Development Server

- allocates central control blocks,
- opens the datasets `STGTRACE`, `STGSTDO`, `STGSTDE`, `STGCONFG`,
- obtains the configuration file,
- loads the Natural front-end,
- initializes the first Natural session and
- launches the TCP/IP listener task.

If one of these steps fails, the server will not be able to continue and will terminate immediately.

Analyze the trace output (`STGTRACE`) or the error output (`STGSTE`) to find out the problem.

`STGTRACE`, `STGSTDO`, `STGSTDE` are synonyms for *`serverid`*`E`, *`serverid`*`O` and *`serverid`*`T`.

## Which dataset should I analyze to get error information?

| | |
|---|---|
| STGSTE | Contains only error output. Each record consists of 2-4 lines, depending on whether it is a Natural error, a system error or an NDV stub error.<br>■ Natural Error<br>1. DayOfMonth Time TaskId UserId<br>2. TaskId NDV Error: error classification<br>3. Natural FrontEnd error or Natural runtime error<br>4. Natural error text<br>■ System Error<br>1. DayOfMonth Time TaskId UserId<br>2. TaskId NDV Error: error classification<br>3. TaskId Sys Error: System error text<br>■ NDV Stub Error<br>1. DayOfMonth Time TaskId UserId<br>2. TaskId NDV Error: error classification |

| | |
|---|---|
| STGTRACE | Contains NDV trace information and error information.<br><br>Each trace record contains DayOfMonth Time TaskId trace information text.<br><br>The string PrintError in the trace information text prefixes errors. |
| STGSTO | Content of the configuration file allocated to STGCONFG. |
| SYSOUT | Messages from LE runtime system. |

## Trace output shows: Cannot load Natural front-end ...

The Natural front-end specified by the NDV configuration parameter `FRONTEND_NAME` was not found in the load library concatenation.

## Trace output shows: Transport initialization failed, EDC8115I address already in use

The TCP/IP port number specified by the NDV configuration parameter `PORT_NUMBER` is already in use by another process.

## Trace output shows: Error at: Template runtime connect

When a Natural Development Server initializes, it starts a Natural session using the session parameter(s) defined by the NDV configuration parameter `SESSION_PARAMETER`. The profile definition of the NDV configuration parameter `DEFAULT_PROFILE` is appended. If the initialization of the template session fails, the server terminates immediately. The original error can be found below the message `Error at:Template runtime connect`.

Typical error situations could be:

- No Natural buffer pool defined.
- Natural system file `FNAT` not accessible.
- Natural profile parameter `ITERM=ON` (Session Termination in Case of Initialization Error).
- NDV initial user ID not defined.

## Definitions required in Natural Security

- Each client must be defined in Natural Security (NSC) if the Transition Period Logon flag in NSC is set to `NO`. Otherwise, your **Map Environment** attempt fails with a NAT0873 error.
- You must define an NDV initial user ID (default ID is `STARGATE`) unless you run with Natural profile parameter `AUTO=OFF` (no automatic logon).
- Each user must have either a default library or a private library. Otherwise, your **Map Environment** attempt will fail with a NAT1699 error.
- You must not specify a startup program that executes an I/O statement or stacks a `LOGON`, `LOGOFF` or `RETURN` command, because the program is executed whenever you change the focus to that library within the tree view.
- If you add a new user, you must specify a password for this user. Otherwise, his/her **Map Environment** attempt will fail with a NAT0838 error.

## Map Environment fails with a NAT3048

Specify session parameter `ETID=' '`. If you are using Natural Security, clear the ETID (Adabas User Identification) definition for that user.

## Map Environment fails with Stub RCnn

Stub return codes are raised by the NDV front-end stub, if it detects a logical processing error when dispatching the NDV request. The NDV trace output contains detailed information about the reason for the error.

The following stub return codes are possible:

| Code | Meaning, Reason, Action |
|---|---|
| 1 | `Error during session reconnect` (for future use). |
| 2 | `Cannot create new session directory entry or subtask.`<br><br>When Natural Studio executes a **Map Environment** command, the Natural Development Server allocates an entry in its session directory and creates a new subtask. If one of these actions fails, the Stub `RC 2` is raised. |
| 3 | `Cannot initialize new session.`<br><br>This error occurs if a storage allocation for internal NDV control buffers fails due to a lack of virtual memory above 16 MB. |

| Code | Meaning, Reason, Action |
|---|---|
| | **Reason:**<br><br>Virtual memory above 16 MB too small.<br><br>**Action:**<br><br>Increase the virtual memory above 16 MB, decrease the number of physical storage threads, distribute the clients to several Natural Development Servers. |
| 4 | `Session execution failed.`<br><br>Internal error. Natural Studio uses an invalid session identifier to process a request.<br><br>**Reason:**<br><br>■ When a **Map Environment** command is issued, the session ID already exists.<br>■ The Natural session with the specified ID is not initialized.<br><br>**Action:**<br><br>Locate the defective session ID in the server trace file and **cancel** it using the **monitor task**, or restart your Natural Studio session. |
| 5 | `I/O execution not allowed.`<br><br>In some situations, a Natural I/O is prohibited at the Natural Development Server.<br><br>**Reason:**<br><br>■ I/O execution during `LOGON` request.<br>■ I/O execution while a transaction processor is executing.<br><br>**Action:**<br><br>Locate the I/O buffer in the server trace file to find out which I/O should be processed. Check for any startup program specified for the library you want to logon to. |
| 6 | Not applicable. |
| 7 | `Error during I/O execution.`<br><br>The Natural Development Server cannot finish a terminal I/O.<br><br>**Reason:**<br><br>■ Virtual memory above 16 MB too small.<br>■ I/O reply buffer sent by Natural Studio is invalid.<br><br>**Action:**<br><br>Increase the virtual memory above 16 MB. If the I/O reply buffer is invalid, contact Software AG support. |
| 8 | `Protocol element missing.` |

| Code | Meaning, Reason, Action |
|---|---|
| | Internal error, contact Software AG support. |
| 9 | `NDV not installed on Natural system file.`<br><br>Natural Development Server cannot execute the Natural module `TRPRO` located on library `SYSLIB`.<br><br>**Reason:**<br><br>■ The NDV modules were not loaded on the system file `FNAT`.<br><br>**Action:**<br><br>Use the Natural utility INPL to load the NDV modules. |
| 10 | `LOGON command required.`<br><br>If you execute a program on the Natural Development Server that executes a `LOGOFF` (or a `RETURN` when no `SETUP` record is available), the logon library is undefined.<br><br>In an online environment, the Natural Security logon screen is displayed in this situation. Under NDV, the Natural session rejects all requests except a `LOGON` command. This applies only if Natural Security is installed. You can issue a `LOGON` command either via the command line or by clicking on any library in your tree view. |

## Special characters are not translated correctly

The ASCII-to-EBCDIC translation for NDV uses the Natural translation tables `TABA1/TABA2`. These tables are automatically or manually adapted at the customer's site.

**Automatic Adaptation of Translation Tables**

Automatic adaptation of the Natural translation tables `TABA1/TABA2` takes place if the following Natural profile parameters are set:

- `CFICU=ON` and
- `CP=`*`value`*

  where *`value`* can be any value except `OFF` or `AUTO`.

For further information on possible settings, see the corresponding profile parameter descriptions in the Natural *Parameter Reference* documentation.

At session initialization (when you map to the NDV server) Natural automatically adapts its conversion tables `TABA1/TABA2` according to the `CP` parameter definition and the code page used at the client. To verify if the conversion tables have been adapted, set NDV `TRACE_LEVEL=31`, connect to the NDV host via Natural Studio, and review the NDV trace file.

Each Map Environment starts with:

```
11 07:58:02 00000003 Got new connection
```

some lines down you find:

```
11 07:58:02 00000005 Client codepage: windows-1252
11 07:58:02 00000005 Client operation = 18
```

and again some lines down you find:

```
11 07:58:03 00000005 TABA1/TABA2 adapted according CP definitions
```

which indicates that the table has been adapted.

**Manual Adaptation of Translation Tables**

The translate tables can be modified as follows:

1. Modify source member `NTTABA1/NTTABA2` on the Natural distribution library. Reassemble `NATCONFG` and relink the Natural nucleus.
2. Specify the Natural session parameter `TABA1/TABA2`.

Manual adaptation requires setting `CP=OFF`. It also requires that `TERMINAL_EMULATION=WEBIO` be off. As a result, you cannot use the statements `REQUEST DOCUMENT` and `PARSE`.

Automatic and manual adaptation are mutually exclusive. If the automatic adaptation is effective, any `TABA1/TABA2` definitions are discarded. You can use either the automatic or the manual update but not a mix of both.

Do not use Natural Studio session parameters as an approach to permanently implementing these changes. You run the risk that different clients may use different translations, and this could corrupt source code the clients share. It is better to maintain the translation centrally. You can do this in two different ways:

1. Maintain the Natural parameter module, or
2. Use the NDV configuration parameter `SESSION_PARAMETER`.

   This affects the SPoD users only.

## Characters are not displayed correctly in the terminal emulation of Natural Studio

In Natural Studio, see also Tools / Options / Workspace / Terminal emulation setting in Natural Studio. The default (Latin) may not be the correct choice. For instance, in the US, you probably want to select “United States”.

A simple Natural program on the mainframe can reveal the EBCDIC representation of a character which is not converting correctly:

```
#A(A1) = '{'
WRITE #A(EM=H)
END
```

If none of the available code pages applies to your needs, it is possible to adapt one of the `N3270_USER 3270` translation tables in the `etc` directory. Details are in the *Natural for Windows* product documentation.

The web-site ***http://www.tachyonsoft.com/uc0000.htm*** is a good resource for finding valid EBCDIC and ASCII values for a given character (glyph) in various code pages.

## How do I find out which hexadecimal value must be specified for TABA1/TABA2?

Run the following program on your Natural for Windows locally.

```
#A(A1) = '{'
WRITE #A(EM=H)
END
```

Output is `7B`.

Run the program on a mainframe (edit the program with the Natural mainframe editor). Output is `75`, assuming that you use a German EBCDIC table. If you use a US EBCDIC table, the output will be `C0`.

Start your Natural Development Server session with `TABA1=(75,7B)` and `TABA2=(7B,75)`.

## The modifications of TABA1/TABA2 do not apply to sources listed in the remote debugger

Specify the NDV configuration parameter `DBG_CODEPAGE=USER`.

## Accessing work files

This topic is discussed in the Natural *Operations for Mainframes* documentation.

## Are there any Natural profile parameter settings required for NDV?

The following Natural profile parameter values are required for NDV:

- `ETID=OFF` is required to allow multiple Natural sessions for each client.
- `DBCLOSE=ON` is required to remove database resources immediately after session termination rather than to keep them until they are removed due to a timeout.
- `ITERM=OFF` is required to continue with the Natural Development Server initialization, even if session initialization errors occur.
- `AUTO=ON/AUTO=OFF` (Automatic Logon) has a different behavior under Natural Single Point of Development. In an online Natural environment, this parameter controls whether you are prompted for your user ID and password or whether your user ID is treated to be a trusted user ID from the TP environment. With Natural Single Point of Development, you must always specify your user ID and password in the Map Environment dialog.

## NAT9915 GETMAIN for thread storage failed

The Natural front-end cannot allocate the Natural thread. Increase the SMARTS SYSPARM parameter `THSIZEABOVE`.

## The NDV server consumes a lot of CPU time even if only a few clients are using it

If you run your NDV server without a CPU time limit on session level, a Natural program might run into an endless loop. Issue a server command `list sessions` and examine whether any of the listed sessions has the status code "IO" (under the column header St. in the list output). The character I means that the client owns an initialized session, and the O flags mean that the client occupies a thread and is currently executing.

If a second `list sessions` command results in an "IO" for the same client with an unaltered Last Activity, it is probably a stuck or looping client. You can try to cancel the session using a `CANCEL SESSION` server command. If the cancelation fails, a restart of the NDV server is required.

If the `list sessions` function does not show a stuck or looping client, cancel the NDV server by using the `DUMP` option, and consult Software AG support.

## I get a NAT0873 internal error at user authentication for Map Environment

Please check the NDV trace file for the message `ExtMsg:Security system not activated (SYSPARM SECSYS)`. This message indicates that SMARTS cannot invoke the external security system specified by the `SYSPARM` parameter `SECSYS`.

## The server fails to start with return code 4 and in the error log I find ‘Transport initialization failed‘

Probably the TCP/IP environment is in error. See the system error message after the error log entry and ask your system programmer(s) for assistance.