Natural Development Server

Natural Development Server for BS2000

Version 8.3.4

October 2017

ADABAS & NATURAL

This document applies to Natural Development Server Version 8.3.4 and all subsequent releases.

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

Document ID: NDV-NNDVDOC-834-20171016

Table of Contents

Preface

This documentation applies to Natural Development Server (product code NDV) under SMARTS on BS2000.

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

Natural Development Server for BS2000 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 in the runtime environment SMARTS on BS2000.
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 Introducing Natural Development Server

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 mainframe environment running under the operating system BS2000.

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

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. **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.

2 Development Server File

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.

3 Natural Development Server on Mainframes

This chapter describes the concept and the architecture of the Natural Development Server (product code NDV) which is designed for use under SMARTS on BS2000.

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:

- **SMARTS runtime environment**

  SMARTS is used to implement a server runtime environment for the execution of the NDV server.

- **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.

NDV under SMARTS on BS2000

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

SMARTS on BS2000 Basics

SMARTS implements a server runtime environment for the execution of the NDV server. Technically, SMARTS represents a C runtime system and implements a nearly full-blown POSIX system. It drives a family of tasks which either process dedicated functionality or process the application payload in parallel-executed worker tasks. The tasks with a dedicated functionality are the main or oc task, the system thread loop task (started as second task), the socket communication task, the pfs task and the sequential file IO task. The pfs task is optional; it processes all IO operations on the POSIX file system (PFS). The PFS is used for NDV work/print file access method, thus allowing the testing of programs which execute access to work or print files.

SMARTS offers a configurable set of resources to process the application workload. These resources are mainly threads and (worker-) tasks. The actual workload is scheduled by the SMARTS kernel using these resources. In case of momentary shortages of one or the other resource, SMARTS is able to queue incoming requests and to roll-out inactive threads.

All data, processed by SMARTS or a SMARTS application, is located in one common memory pool, the data pool. All code modules, except for some smaller bootstrap routines, are loaded as shared code into another common memory pool, the code pool.

The worker-tasks are the processes (TSNs) by which the Natural runtime is executed. The amount of storage requested by Natural is located in the SMARTS threads within the data pool during the execution of a transaction. If the number of sessions to be processed exceeds the number of threads defined, an internal facility is invoked for rolling the threads. Threads rolled out are placed in compressed format in a so-called roll bufferpool which resides in the data common memory pool as well.

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 front-end is called (together with the Natural runtime system) by the front-end stub for session initialization/termination, request execution and session roll-in/roll-out.

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

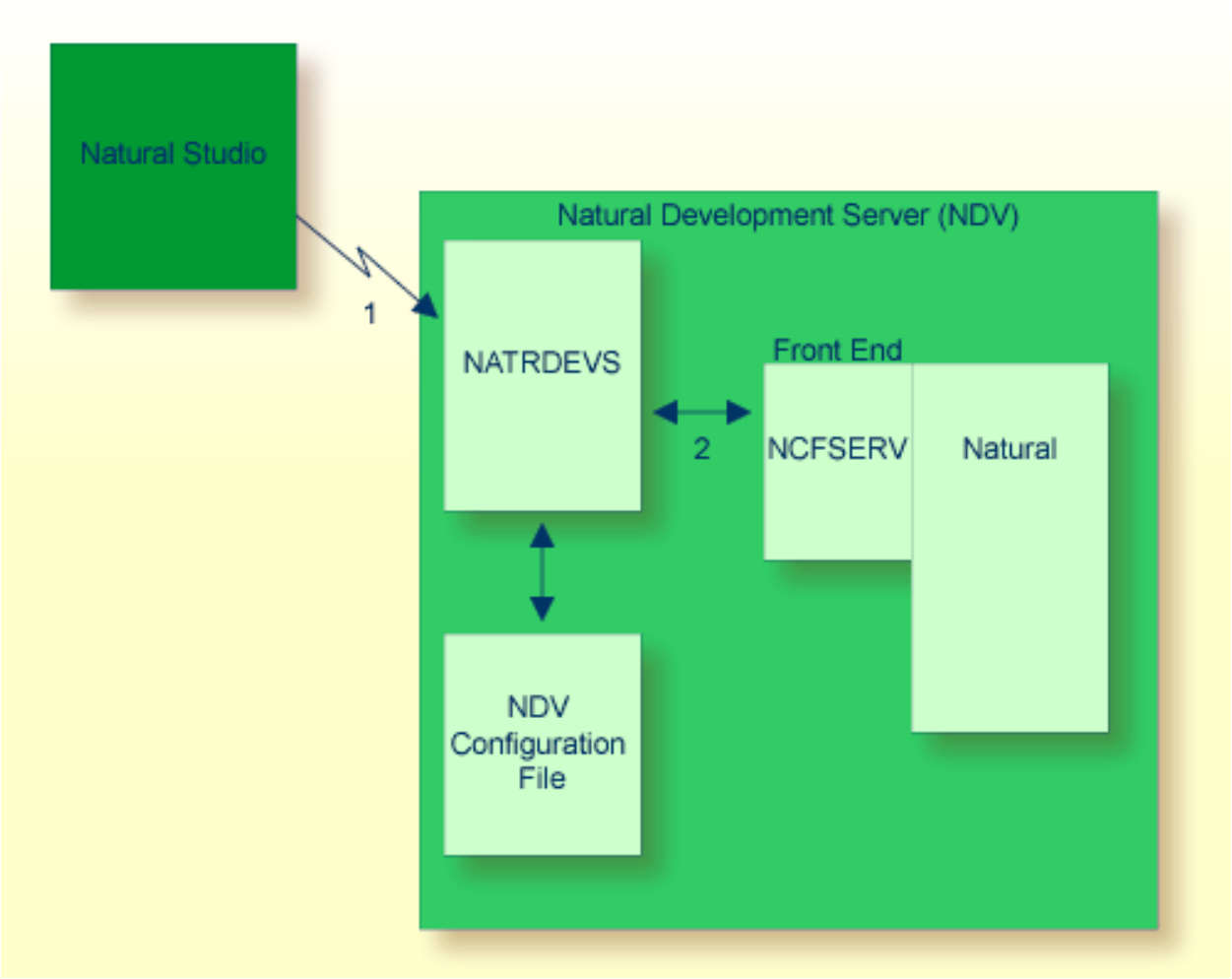
The gateway module `NATGWSTG` is already included in the Natural nucleus.

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).

4 Prerequisites

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 ; see *Configuring the Natural Development Server*, *SMARTS SYSPARM Parameters*, `SERVER` parameter description.

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

  If you are installing without SMA, see *Installation for BS2000*, *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 8.3 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 under SMARTS on BS2000

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

- BS2000 must have been installed on the server mainframe.

  Version as specified under *Operating/Teleprocessing Systems Required* in the current *Natural Release Notes for Mainframes*.
- SMARTS must be installed (product code APS).
- The Natural Com-plete/SMARTS Interface must be installed (product code NCF).

5 Installing the Natural Development Server

This chapter describes how to install a Natural Development Server (product code NDV) in the runtime environment SMARTS on BS2000.

Prerequisites

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

Installation Medium

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
APS*vrs*.LIB	Contains the load modules and the procedures of the SMARTS server.
APS*vrs*.LO*pp*	Patch-level of SMARTS (product code APS). The content of this library must be copied into the library APS*vrs*.LIB.
NDV*vrs*.MOD	Contains the load modules of the Natural Development Server.
NDV*vrs*.JOBS	Contains the Installation Job Control for those customers who want to install without using System Maintenance Aid (SMA).
NDV*vrs*.INPL	Contains the transaction processor, the error messages of the transaction processor, and the sample programs required for using the tutorial. See *First Steps with Natural Single Point of Development* in the Natural Single Point of Development documentation.
NDV*vrs*.SYSF	Contains the field definition table (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 corresponding *Natural Release Notes for Mainframes*).
NCF*vrs*.MOD	Contains the load modules of the Natural Com-plete Interface (required for SMARTS).
NCF*vrs*.MAC	Contains the macros of the Natural Com-plete Interface (required for SMARTS).

- where

vrs in a dataset name represents the version, release and system maintenance level of the product.

pp in a dataset name represents the patch level of the product.

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

Content of the Development Server JOBLIB

Naming Conventions

In the following text, the library name JOBLIB stands for

- the example job library (NDV*vrs*.JOBS), if you are not using SMA, or
- the SMA job library (see SMA parameter JOBLIB in SMA Parameter Group BASIC), if you are using SMA.

Software AG uses the following naming conventions for source elements in the library JOBLIB:

A<*product-code*><*function*> = Assembler sources

L<*product-code*>< *function*> = Instruction for TSOSLNK/BINDER

Important Elements of the Job Library

Element	Description
NDV-SYSPARM	SMARTS parameters for NDV. See also ***SMARTS SYSPARM Parameters***.
NDV-CONFG	NDV configuration parameters. See also ***NDV Configuration Parameters***.
NDV-ADAPARM	ADALNK parameter for NDV.
ANDVPARM	Assembler source - Natural parameter module for NDV.
LNATSHAR	Link instructions to link the Natural nucleus.
LNDVFRNT	Link instructions to link the NDV front-end module.
START-NDV	Procedure to start the NDV server.
STOP-NDV	Procedure to stop the NDV server.
SHOW-SYSOUT	Procedure to show the SYSOUT file of the FSIO task of NDV.

Installation Procedure

To install the Natural Development Server in a SMARTS environment on BS2000, perform the following steps:

- Step 1: Load FDIC system file
- Step 2: Assemble the Natural parameter module
- Step 3: Relink the Natural nucleus module
- Step 4: Link the NDV front-end module
- Step 5: Load Natural objects, error messages and samples for NDV
- Step 6: Create a PFS for print file/work file support
- Step 7: Copy DDMs and processing rules to FDIC

Step 1: 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 by using the dataset `NDV`*`vrs`*`.SYSF`.

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

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 2: Assemble the Natural parameter module

(Job I055, Step 8420)

Assemble the source `ANDVPARM` which is contained in the library `JOBLIB`.

Step 3: Relink the Natural nucleus module

(Job I060, Step 3802)

This step is optional.

The Natural nucleus module must be relinked with the modules `NAT3270` and `NAT3279` if they have not been included yet. This is accomplished by using the source `LNATSHAR` which is contained in the library `JOBLIB`.

If you are using SMA, please refer to the SMA read-me file of `NDV`*`vrn`* (skeleton `##NDV`*`vrn`*`-README`).

Step 4: Link the NDV front-end module

(Job I060, Steps 8410)

Link the NDV front-end module by using the source `LNDVFRNT` which is contained in the library `JOBLIB`.

Step 5: Load Natural objects, error messages and samples for NDV

(Job I061, Steps 8450)

During the loading of the NDV related objects with the Natural utility INPL, the assigned system file(s) FDIC and/or FSEC is/are initialized with NDV-specific information.

- Using the INPL utility, load the objects from dataset NDV*vrs*.INPL onto your Natural system file (FNAT). The Natural profile parameter FDIC must have been set to point to your development server file.
- The error messages are loaded.
- The sample programs to use the tutorial (see *First Steps with Natural Single Point of Development*), are loaded to your Natural system file (FNAT).

Step 6: Create a PFS for print file/work file support

(Job I085, Step 8400)

The procedure CREATE-PFS is called to allocate and format a PFS file.

This procedure has to be supplied with the following parameters:

FILE-NAME	The name of the container file.
FILE-SIZE	The size of the container file. To be specified either in K for kilobytes or M for megabytes.
APS-LIB	The SMARTS library where the format utility PFSUTIL resides.

See also ***PFS Description*** and ***Allocating and Configuring a SMARTS Portable File System*** at the end of this document.

Step 7: 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 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 (NSC) with a valid default library. Refer also to NDV configuration parameter `INITIAL_USERID` in the section *Development Server Configuration*. Alternatively, you can define 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 a remote development client is not defined, the **Map Environment** dialog will return a Natural Security (NSC) error.

Make sure that a user who is defined in Natural Security has a default library or a private library defined. Otherwise, If he/she logs on to the server from an NDV client, an error message NAT0815 will occur.

Step 9: Debug a Natural batch job with Debug Attach for NaturalONE

Sample JCL - ISP Format

```
/LOGON
/SYSFILE SYSOUT=ATDEBUG.OUT
/SYSFILE SYSLST=ATDEBUG.LST
/FILE ADAPARM,DDLNKPAR
/FILE NATvrs.MOD,LINK=BLSLIB01
/FILE NDVvrs.MOD,LINK=BLSLIB02
/FILE CMPRMIN.RMDBG,CMPRMIN
/FILE ADAvrs.MOD,DDLIB
/FILE DBGTRACE.NATBATCH,LINK=DBGTRACE,FCBTYPE=ISAM,SHARUPD=YES    1                ↩

/START-EXE-PROG F-F=*LI-E(L=NATvrs.MOD,EL=NATvrsBA,TYPE=L)
/LOGOFF
```

Sample JCL - SDF Format

```
/SET-LOGON-PARAMETERS USER-IDENTIFICATION=*NO,ACCOUNT=*NONE
/ASSIGN-SYSOUT TO=ATDEBUG.OUT
/ASSIGN-SYSLST TO=ATDEBUG.LST
/CREATE-FILE FILE-NAME=ADAPARM,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=DDLNKPAR,FILE-NAME=ADAPARM,SUPPORT=*DISK
/CREATE-FILE FILE-NAME=NATvrs.MOD,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=NATvrs.MOD,SUPPORT=*DISK
/CREATE-FILE FILE-NAME=NDVvrs.MOD,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=NDVvrs.MOD,SUPPORT=*DISK
```

```
/CREATE-FILE FILE-NAME=CMPRMIN.RMDBG,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=CMPRMIN,FILE-NAME=CMPRMIN.RMDBG,SUPPORT=*DISK
/CREATE-FILE FILE-NAME=ADAvrs.MOD,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=DDLIB,FILE-NAME=ADAvrs.MOD,SUPPORT=*DISK
/CREATE-FILE FILE-NAME=DBGTRACE.NATBATCH,SUPPRESS-ERRORS=*FILE-EXISTING
/ADD-FILE-LINK LINK-NAME=DBGTRACE,FILE-NAME=DBGTRACE.NATBATCH,SUPPORT=-
/     *DISK(SHARED-UPDATE=*YES)                    1                                ↩

/START-EXE-PROG F-F=*LI-E(L=NATvrs.MOD,EL=NATvrsBA,TYPE=L)
/EXIT-JOB MODE=NORMAL
```

[1]: Specifies the trace file. The `SHARED-UPDATE` option enables the user to lookup the trace file while the debug task is still running.

The Natural parameter file `CMPRMIN.RMDBG` for example has the following content:

```
DBGAT=(ACTIVE=ON,CLID=CLIENT,HOST=DASSERV,PORT=2500),
RCA=NATATDBG
```

Important: If the Natural environment dependent nucleus is shared by various Natural applications that are running under TIAM, UTM or batch, NATAD*vrs* has to be linked to each of the Natural environment dependent nuclei. For the assemblies of the Natural drivers this means, that the parameter `LINK=(NATADBG)` has to be specified to enable the BS2000 loader to find the load points of NATAD*vrs* in the particular environment dependent nuclei.

Installation Verification

Map Environment Dialog

For information on how to map an environment, refer to the Natural Single Point of Development documentation, section *First Steps with Natural SPoD*, *Local and Remote Environment*, *Connecting to a Development Server for the First Time*.

Check Messages

BS2000 Operator Console

```
APSSVR0026-* Server NCFNAT41 started
APSOPC0000-* SMARTServer is initialized
```

SYSOUT of FSIO-Task

The following messages are displayed when parameter `TRACE_LEVEL=31` was set:

```
Template session initialized
Template runtime connected
Template runtime terminated
Template ready for clone
Main listener activated
Setup PAL environment successful
Listener thread launched 01625028
**********************************************
************ Server is up **************
**********************************************
Waiting for terminate event...
```

SMARTS Portable File System on BS2000

- PFS Description
- Allocating and Configuring a SMARTS Portable File System

PFS Description

SMARTS PFS (Portable or POSIX File System) implements a file system, known from UNIX systems, in a mainframe environment. Basically, it consists of a container file, which comprises all (UNIX) files and a corresponding (logical) access method, which processes all requested I/O operations.

The container file has to be allocated and preformatted using the BS2000 procedure `CREATE-PFS` (described below).

The PFS maps all file names to a node of a directory tree within the physical container file. In the case of BS2000, this container file is a PAM file with a block size of 4 KB (`STD,2`).

Within Natural, the actual path is specified by a corresponding `DEFINE WORK FILE` statement in the program which executes work file or print file access.

Each node (subdirectory) is separated by a slash (/) from its parent. The highest level qualifies the file name.

Example 1:

```
DEFINE WORK FILE 1 '/MISC/USER1/TESTFILE/'
```

Specifies: `ROOT' => MISC => 'USER1 => 'TESTFILE`

`://MISC/USER1/TESTFILE`

Example 2:

```
DEFINE WORK FILE 1 'TESTFILE2.W01'
```

Specifies: `ROOT' => TESTFILE2.W01`

Allocating and Configuring a SMARTS Portable File System

The Natural server uses the SMARTS portable file system (PFS) as a data container for Natural work files, print files, temporary sort files and the editor work file. The SMARTS PFS is the only storage medium available for these files under SMARTS.

In order to be able to use the PFS for Natural files you have to configure Natural accordingly:

- For work files or print files, specify the access method `AM=SMARTS`, using the Natural profile parameters `WORK` (Work-File Assignments) and/or `PRINT` (Print File Assignments).
- For temporary sort files, specify the type of storage medium `STORAGE=SMARTS`, using the Natural profile parameter `SORT` (Control of Sort Program).
- And to allocate the editor work file in the PFS, specify the work file mode `FMODE=SM`, using the Natural profile parameter `EDBP` (Software AG Editor Buffer Pool Definitions).

If you use one of these options, you have to configure your SMARTS to use a PFS.

Step 1: Allocate a PFS

The file PFS.TST should not exist before the allocation procedure is executed, otherwise DMS error DMS0683 will occur.

The initialization can be done by using the following procedure:

```
/CLP FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=APSvrs.LIB,ELEMENT=CREATE-PFS, -
/   TYPE=SYSJ),PROCEDURE-PARAMETERS=(FILE-NAME=PFS.TEST,SIZE=4096K,        -
/   APS-LIB=APSvrs.LIB) ↩
```

Step 2: Configure SMARTS

In the SMARTS `SYSPARM` configuration file, add the following lines:

```
CDI_DRIVER=('CIO,PAAQBIO,PFSTSK=PFSTASK,TRACE=N')
CDI_DRIVER=('TESTPFS,PAANPFS,LRECL=4096,CONTAINER=CIO:PFS/TEST/')
MOUNT_FS=('TESTPFS://','/')
```

6 Configuring the Natural Development Server

This chapter describes how to configure a Natural Development Server that is running in a SMARTS environment on BS2000.

SMARTS Configuration File

The Natural Development Server reads its configuration parameters from a file which resides as an S-type member in the NDV-JOBS library.

Notes:

1. Translation tables are used to convert characters when sending or receiving data to or from a host while working with a terminal emulation, see *Adapting Translation Tables for Natural Development Server under BS2000* in the Natural for Windows (Natural Studio) documentation.
2. The SMARTS configuration is contained as an S-type member which resides in the NDV-JOBLIB. It has to be passed to the system in the startup procedure parameter NDV-SYSPARM.

SMARTS SYSPARM Parameters

The Natural Development Server requires the following SMARTS SYSPARM parameters:

Parameter	Definition	
RESIDENTPAGE	This parameter specifies one ore more programs which are loaded into the SMARTS load-pool during system startup. The following members must be defined in the SMARTS resident area: NATRDEVS, NATSOCK, NATMONI, Natural front-end (NCFNUC) and Natural nucleus.	
SERVER	The following SERVER definitions are required for the Natural Development Server:	
	SERVER=(POSIX,PAENKERN,CONF=PAANCONF)	The POSIX server.
	SERVER=(NCFNAT*nn*,NCFNAT*nn*)	The Natural bufferpool server. The size of the various Natural bufferpools is configured in the parameter string of the NDV configuration parameter SESSION_PARAMETER.
	SERVER=(NDVSERV,PAENAPPS,NATRDEVS,'*SERVERID*')	The NDV server *SERVERID*.
CDI_DRIVER	The CDI (Communication Driver Interface) drivers specify the various I/O routines and/or tasks of SMARTS as well as the physical access paths for sequential files and	

Parameter	Definition	
	PFS-container files. If necessary for technical reasons, these routines are implemented as separate tasks (socket communication, physical file I/O, etc.). The others are executed in the oc-task (main task) or in one of the worker-tasks (for example, console I/O).	
	`CDI_DRIVER=('console,PAANCNIO')`	The console server.
	`CDI_DRIVER=('file,PA2SFIO,SIOTSK=`*`JOBNAME`*`')`	The file I/O task *JOBNAME* (SYSOUT, traces, etc.).
	`CDI_DRIVER=('tcpip,PAAZSOCK,SOCKTSK=`*`JOBNAME`*`')`	The socket task *JOBNAME*.
	`CDI_DRIVER=('CIO,PAAQBIO,PFSTSK=')`	The PFS task *JOBNAME* running the CDI driver CIO.
	`CDI_DRIVER=('PFS,PAANPFS,LRECL=4096, CONTAINER=CIO:AAA/BBB/CC')`	Defines the container file `$USERID.AAA.BBB.CC` for the Portable File System (PFS); see also ***SMARTS Portable File System on BS2000***.
`MOUNT_FS`	`MOUNT_FS=('`*`PFS`*`://','/')`	Mapping of (PFS) file names to the appropriate physical BS2000 container file; see also ***SMARTS Portable File System on BS2000***. This parameter maps (POSIX) filenames to a physical BS2000 container file for the specified PFS.
`DATA-MAXIMUM`	`DATA-MAXIMUM=`*`nnnn`	Maximum possible CMP size for `DATA` in MB (over all sessions). This parameter limits the maximum size of the data common memory pool. At SMARTS startup, the data CMP is enabled with the specified size, however, real storage allocations within the pool are done on demand only in the actually requested size (For more information, please refer to the BS2000 executive macros manual: *ENAMP / REQMP macros*).
`CODE-MAXIMUM`	`CODE-MAXIMUM=`*`nn`	Maximum possible CMP size for (shared) Code in MB. This parameter limits the total size of all routines loaded, such as the SMARTS kernel itself, NDV and the Natural nucleus.
`THREAD-GROUP`	`THREAD-GROUP=(DEFAULT,(DEFAULT,0,`*`n`*`)`	Establishes the default thread-group with *n* threads. Only the *num* subparameter is of importance. This

Parameter	Definition	
		value defines the number of sessions that can be active and kept in storage in parallel; see also the WORKLOAD and TASK-GROUP parameters.
TASK-GROUP	TASK-GROUP=(DEFAULT,*n*)	Specifies the number of active worker-tasks. The value *n* should correspond to the specified number of threads. Only the *num* subparameter is of importance. This value defines the number of worker-tasks (BS2000 tasks) that can execute NDV Natural sessions in parallel. Evidently, this parameter value should correspond to the number of available threads as defined by the THREAD-GROUP definition!
THSIZEABOVE	THSIZEABOVE=*nnnn*	Specifies the SMARTS thread size (in KB) which has to contain all buffers of one Natural session. This parameter specifies the size of the SMARTS threads which have to contain the Natural thread (NTHSIZE parameter, defined in NCFPARM). A certain headroom (20 % or more, depending on your environment) is recommended. If the Natural Development Server initialization fails with NAT9915 GETMAIN for thread storage failed, this parameter must be increased.
WORKLOAD-AVERAGE	WORKLOAD-AVERAGE=*n*	The average number of sessions active in parallel. This parameter defines the expected average number of Natural sessions (not users, since one user can start more than one session!) which are to be executed by the server. This parameter must not be confused with the THREAD-GROUP parameter, because it represents the sum of all active and inactive sessions.
WORKLOAD-MAXIMUM	WORKLOAD-MAXIMUM=*nn*	The maximum possible number of sessions active in parallel. This

Parameter	Definition	
		parameter defines the maximum possible number of Natural sessions.
ADASVC	ADASVC=249	The Adabas SVC number must not be changed.

Note: The parameter values printed in italics (*SERVERID*, *JOBNAME*, *PFS*, *nnn*, etc.) are to be specified by the user.

SMARTS Server Environment Configuration Parameters

The following general parameter descriptions are adapted excerpts from the original SMARTS documentation. The text is provided for background information only. Therefore, not all of the information contained therein applies to the Natural Development Server under SMARTS on BS2000.

- ADASVC
- RESIDENTPAGE
- SERVER
- TASK-GROUP
- THREAD-GROUP
- THSIZEABOVE
- WORKLOAD-AVERAGE, WORKLOAD-MAXIMUM

ADASVC

This parameter is for internal use only. Do not change the Adabas SVC number.

RESIDENTPAGE

Sysparm	Use	Possible Values	Default
RESIDENTPAGE	The name of a program to be loaded and made resident when SMARTS is initialized.	*program-name*	none

All modules are assumed to be reentrant, and are loaded into the address space automatically at first reference.

The program must be fully reentrant. If it is not marked reentrant, a warning message is issued on the operator's console at SMARTS initialization time.

The program must reside in the APS-LIB or in the NDV-MOD library of the SMARTS initialization procedure.

SERVER

Sysparm	Use	Possible Values	Default
SERVER	Information that identifies a server to SMARTS.	*server-information*	none

The *server-information* has the format

```
(serv-id,init-mod,p1,p2,pn)
```

- where

serv-id	is the ID for this server (1-8 characters).
init-mod	is the name of the initialization/termination routine.
p1,p2,pn	are parameters to be passed to the initialization routines.

Specifying the SERVER parameter causes SMARTS to build a server directory entry (SDE) for the specified server and pass control to the initialization routine specified to initialize the server.

TASK-GROUP

Sysparm	Use	Possible Values	Default
TASK-GROUP	A group comprising one or more tasks, available when SMARTS is started.	(*grp*, *num*, *priority*, *maxq*)	(DEFAULT, *num*)

- where

grp	Required. The name of the task group being defined. The default task group is DEFAULT.
num	Required. The number of tasks to be allocated in the task group. The default number of tasks is calculated dynamically based on the size of the installation.
priority	Not supported under BS2000.
maxq	The maximum number of TIBs (default 16) expected on this task group's work queue at the same time. Under normal circumstances, the default should be adequate. When there are problems and it is not, a secondary Last In First Out (LIFO) queue is used so that no work is lost. The normal queue is First In First Out (FIFO), which ensures that work is done in the order in which it is received. This is why the LIFO queue is only used as a secondary backup.

Important: For SMARTS, only the task group DEFAULT is available. Software AG strongly recommends that you use the default definition. If other products running on SMARTS require changes to the defaults or allow the definition of their own task groups, that will be indicated in the relevant documentation.

Notes:

1. A maximum of 8 task groups may be defined.
2. Task-group names are converted to uppercase prior to being processed; therefore, a parameter entered in lowercase is treated as, and appears in, uppercase letters.
3. If more than one specification appears for a task group, the last valid specification is used.
4. The task group `DEFAULT` must always exist in the system. If it is not explicitly defined by the installation, the task group is built by the system with the default values.
5. Note that the total number of tasks to be attached must not exceed the `MAXTASKS` specification. This is not checked until the task groups are being built; however, exceeding the value leads to task-group allocation errors as against parameter errors.

THREAD-GROUP

Sysparm	Use	Possible Values	Default
`THREAD-GROUP`	A thread group containing one or more thread subgroups and threads, to be available when SMARTS is started.	see below	see below

The format for the value is

```
(grp,(sub,size,num,cpu,real,key),...,(sub,size,num,cpu,real,key))
```

- where

grp	Required. The name of the task group being defined.
sub	The name of the subgroup being defined. If a subgroup name is specified more than once for the same group, the last valid specification is used when parameter processing has been completed.
size	Required. The amount of storage in kilobytes to be allocated for each thread below the line. A valid value is between 8 KB and 1 MB.
num	The number of threads to be allocated in the thread subgroup. The value must be greater than 1 and less than 4096. Generally, this subparameter is required. It can be omitted for one (and only one) thread subgroup in the address space; in this case, the number of threads to be allocated for the subgroup is calculated dynamically by SMARTS, based on the size of the installation.
cpu	The CPU time in seconds (default `0.00`) that a user program can use in the thread subgroup for one SMARTS transaction. This value may be entered as an integer or to a level of hundredths of seconds using the *n.nn* format. If a `0` is provided as the `CPUTIME` for a thread subgroup, no CPU limit is placed on programs running in the associated threads.
real	The wait time in seconds (default `0.00`) for the thread subgroup, after which a message is issued to the console if the user program has not given up control of its thread. This value may be entered as an integer or to a level of hundredths of seconds using the *n.nn* format. If `0` is specified, elapsed time is not checked for the thread subgroup.
key	The key (default `M`) in which the threads within the subgroups are allocated: `M` - The thread keys are a mixture of user keys excluding the key in which SMARTS is running. `N` - No storage protection is implemented and all threads run in the same key as SMARTS.

Note: The user may also specify a value in the range 1 to 15 inclusive to allocate a thread to that key explicitly.

The default value is

```
THREAD-GROUP=(DEFAULT,($DEFAULT,8,num))
```

- where *num* is calculated dynamically based on the size of the installation.

Important: For SMARTS, only the thread group `DEFAULT` is available. Software AG strongly recommends that you use the default definition. If other products running on SMARTS require changes to the defaults or allow the definition of their own thread groups, that will be indicated in the relevant documentation.

Notes:

1. A maximum of 8 thread groups may be defined.
2. A maximum of 8 subgroups can be allocated per thread group. The subgroups may be defined on one line or on different lines. When a second `THREAD-GROUP` statement is used, the same group name must be specified to relate the subgroup entries.
3. Thread group and subgroup names are converted to uppercase prior to being processed; therefore, a parameter entered in lowercase is treated as, and appears in, uppercase letters.
4. If more than one specification appears for a thread subgroup of a thread group, the last valid specification is used.
5. The amount of storage specified on the `THSIZEABOVE` parameter is allocated above the line for each thread defined as a result of the `THREAD-GROUP` parameter.
6. The thread group `DEFAULT` must always exist in the system. If it is not explicitly defined by the installation, the thread group is built by the system with the default values. If it is defined, the system ensures that a thread subgroup with a thread size at least as large as that required by `DEFAULT` is allocated. If not, the system allocates an additional subgroup for the group. If too many subgroups have been defined, the last one defined is overwritten to allow for the default specification.
7. The keyword data is processed from left to right. If more than one thread subgroup is defined on one line and the line contains an error, even if an error message is issued for the line, any subgroups processed up to the error are still accepted. That is to say, if the first subgroup is correct and the second is not, an error message is issued but the first thread subgroup is defined while the second and subsequent specifications in the same statement are ignored.

THSIZEABOVE

Sysparm	Use	Possible Values	Default
THSIZEABOVE	The amount of storage above the 16 MB line, in multiples of 1024 bytes, to be allocated to each thread.	*n*	1024

WORKLOAD-AVERAGE, WORKLOAD-MAXIMUM

The WORKLOAD-AVERAGE parameter specifies a normal workload value, and the WORKLOAD-MAXIMUM parameter specifies a maximum workload value. SMARTS uses these values togther with the region sizes above and below the 16 MB line to configure itself.

These parameters are not required, but tuning them may improve performance.

Sysparm	Use	Possible Values	Default
WORKLOAD-AVERAGE	The average number of parallel processes expected to run in SMARTS.	1-32767	WORKLOAD-MAXIMUM divided by 4.
WORKLOAD-MAXIMUM	The maximum number of parallel processes expected to run in SMARTS.	1-32767	50 if WORKLOAD-AVERAGE is not specified, otherwise WORKLOAD-AVERAGE times 4.

SMARTS Sample Configuration

```
ADASVC=249
DATA-MAXIMUM=160
CODE-MAXIMUM=20
THSIZEABOVE=4096
THREAD-GROUP=(DEFAULT,(DEFAULT,0,4)
TASK-GROUP=(DEFAULT,2)
WORKLOAD-AVERAGE=8
WORKLOAD-MAXIMUM=40
RESIDENTPAGE=NATSOCK
RESIDENTPAGE=NATMONI
RESIDENTPAGE=NATRDEVS
RESIDENTPAGE=NCFSERV
RESIDENTPAGE=NvrLPRRB
**********SERVERS ****************************
SERVER=(POSIX,PAENKERN,CONF=PAANCONF)
SERVER=(NCFNATvr,NCFNATvr,1,2048,2,100,4,2048)
SERVER=(NDVNATv,PAENAPPS,NATRDEVS,'NDVS01')
**********CDI-DRIVERS************************
CDI_DRIVER=('file,PA2SFIO,SIOTSK=HSFSIO')
CDI_DRIVER=('console,PAANCNIO')
CDI_DRIVER=('tcpip,PAAZSOCK,SOCKTSK=HSSOCTA2,TRACE=1')
CDI_DRIVER=('CIO,PAAQBIO,PFSTSK=HSPFS,TRACE=N')
```

```
CDI_DRIVER=('PFS,PAANPFS,LRECL=4096,CONTAINER=CIO:NDV/ROOT/SERVER1/')
MOUNT_FS=('PFS://','/')
```

where *vr* is the current version and release number.

NDV Configuration File

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
- FRONTEND_NAME
- HANDLE_ABEND
- HTPMON_ADMIN_PSW
- HTPMON_PORT
- HOST_NAME
- INITIAL_USERID
- MINIMUM_STUDIO_VERSION
- O4I
- PORT_NUMBER
- 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: *profile-name*,*dbid*,*fnr*,*password*,*cipher-code* **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`.

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=NATvrsSV
```

- 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
```

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 *NDV Clients must be defined to Natural Security* (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
```

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
```

Note: Some port numbers are privileged and reserved for certain system services. Ask your BS2000 system administrator for the port number range available to you.

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 `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. or 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
FRONTEND_NAME=NDVFRONT       # and another comment
PORT_NUMBER=4711
# SESSION_PARAMETER='DU=ON'
# TRACE_LEVEL=15+31
```

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.

7 Operating the Natural Development Server

This chapter describes how to operate a Natural Development Server in a SMARTS on BS2000 environment.

Starting the Natural Development Server

Start the Natural Development Server with the SDF command:

```
/Enter-Proc *LIB(NDVvrs.JOBS,START-NDV)
```

The SDF procedure START-NDV has to be supplied with the following parameters:

Parameter	Definition	Default Value
NDV-JOBS	The NDV (SMA) job library.	NDV*vrs*.JOBS
ENV-MOD	The NDV environment-specific module library. This library contains the linked Natural nucleus module.	NDV*vrs*.JOBS
NDV-MOD	The NDV module library.	$SAG.NDV*vrs*.MOD
NCF-MOD	The Natural Com-plete interface module library.	$SAG.NCF*vrs*.MOD
APS-LIB	The SMARTS library (modules and procedures).	$SAG.APS*vrs*.LIB
ADA-MOD	The Adabas module library.	$SAG.ADA*vrs*.MOD
NDV-SERVERID	NDV Server-ID.	NDVS01
PROC-NAME	The name of the NDV START procedure. The procedure must reside in the NDV-JOBS library.	START-NDV
NDV-CONFG	The NDV configuration file. It must reside in the NDV-JOBS library (Type S).	NDV-CONFG
NDV-SYSPARM	The SMARTS configuration file. It must reside in the NDV-JOBS library.	NDV-SYSPARM
NDV-ADAPARM	The ADALNK parameter file (IDTNAME, etc). It must reside in the NDV-JOBS library.	NDV-ADAPARM
LOG-FILE-PREFIX	The log-file prefix for the SYSOUT files of all SMARTS tasks.	L.NDV.
WORKER-JOB-NAME	The job name of the worker-tasks.	NDVWORK
WORKER-JOB-CLASS	The job class of the worker-tasks.	*STD
WORKER-CPU-LIMIT	The CPU time limit for the SMARTS worker tasks.	*NO
LOGGING	Switches logging for diagnostic purposes.	*NO
MAIN-TASK	For internal use only. Do not modify!	-

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

Caution: Do not modify the variable names and parameter names that are used in the procedure START_NDV. This procedure is called recursively to attach the worker-tasks. For this purpose, the ENTERPARM string is internally executed and several variables are read from the system, using the GETVAR function of SDF-P.

Terminating the Natural Development Server

The Natural Development Server can be terminated with the console command:

```
/INTR <TSN smarts-main-task>,EOJ
```

Changing the SYSOUT File Assignment of the FSIO Task

The central logical system file SYSOUT written by the FSIO task can be reassigned at SMARTS server execution time, using the SDF procedure SHOW-SYSOUT.

Thus it is possible to look up trace outputs, error reports, etc., without having to terminate the server.

The procedure SHOW-SYSOUT has to be called with the following parameters:

APS-LIB	The SMARTS module library.
TSN	The TSN of the SMARTS server main-task.

As a result of the SHOW-SYSOUT execution, the logical system file SYSOUT of the FSIO task will be reassigned to a new file and the previous one will be opened with the SHOW-FILE command. The new logical system file SYSOUT is built by appending a numerical suffix to the file name. The value of the suffix is incremented by 1, each time SHOW-SYSOUT is executed.

Example:

```
/Call-Proc *LIB(NDVvrs.JOBS,SHOW-SYSOUT), <SMARTS-tsn>
```

where *SMARTS-tsn* is the TSN of your SMARTS main task.

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***. Or you can use the HTML Monitor Client that supports standard web browser; see ***HTML Monitor Client***.

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 `I'm still up`
`terminate`	Terminates the server.
`abort`	Terminates the server immediately without releasing any resources.
`set` *`configvariable value`*	With the `set` command, you can modify server configuration settings. For example, to modify `TRACE_LEVEL`: `set TRACE_LEVEL 0x00000012`
`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`.
`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 the logical system file `SYSOUT` of the `FSIO` task.

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.

Support of NATRJE Functionality

Calls to the `NATRJE` utility (Natural Remote Job Entry) are supported. For technical reasons, however, the file name of the job to be submitted is created in a way which is different from batch operation in a TIAM or *open*UTM environment. The file name has the following structure:

```
E.DDMMYYYY.XXXXXXXX.program-name.terminal-id
```

Where:

DDMMYYYY	is the day, month and year when the NATRJE call was submitted.
XXXXXXXX	is the time since midnight (00.00 h) in hundreds of seconds.
program-name	is the name of the Natural program that calls the NATRJE utility for the purpose of submitting JCL cards to be executed by the operating system.
terminal-id	is the Natural terminal ID as contained in the system system variable *INIT-ID.

8 Monitor Client NATMOPI

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.

Prerequisites for NATMOPI Execution

Execute NATMOPI with the following SMARTS console command:

```
/INTR <SMARTS-tsn>,NATMOPI <mopi-command>
```

where *SMARTS-tsn* is the TSN of your SMARTS main task.

Example:

```
/INTR 4711,NATMOPI -dls
```

The output is written to the SYSOUT file of the FSIO task.

Note: The server to be monitored must be running in the same SMARTS environment as NATMOPI.

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. **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

`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.

9 HTML Monitor Client

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.

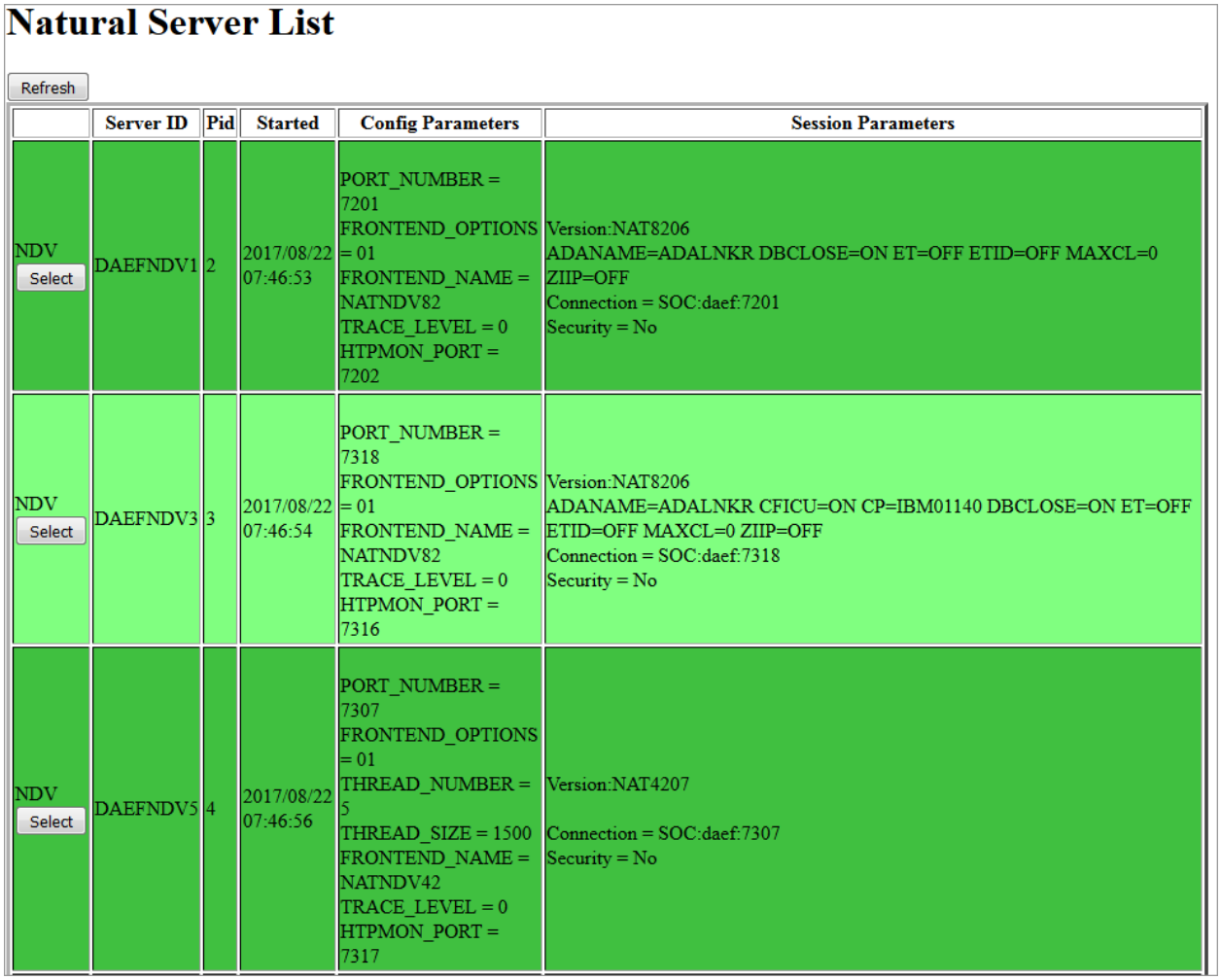
Prerequisites for HTML Monitor Client

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

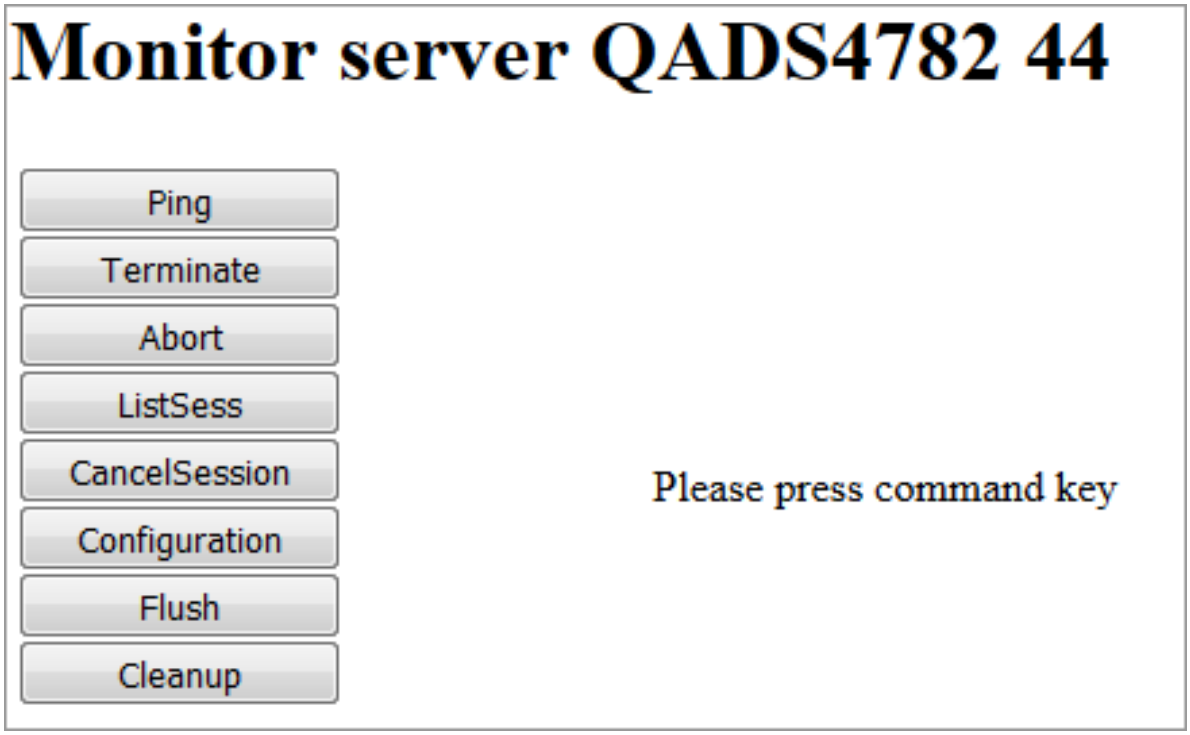
	Server ID	Pid	Started	Config Parameters	Session Parameters
NDV Select	DAEFNDV1	2	2017/08/22 07:46:53	PORT_NUMBER = 7201 FRONTEND_OPTIONS = 01 FRONTEND_NAME = NATNDV82 TRACE_LEVEL = 0 HTPMON_PORT = 7202	Version:NAT8206 ADANAME=ADALNKR DBCLOSE=ON ET=OFF ETID=OFF MAXCL=0 ZIIP=OFF Connection = SOC:daef:7201 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

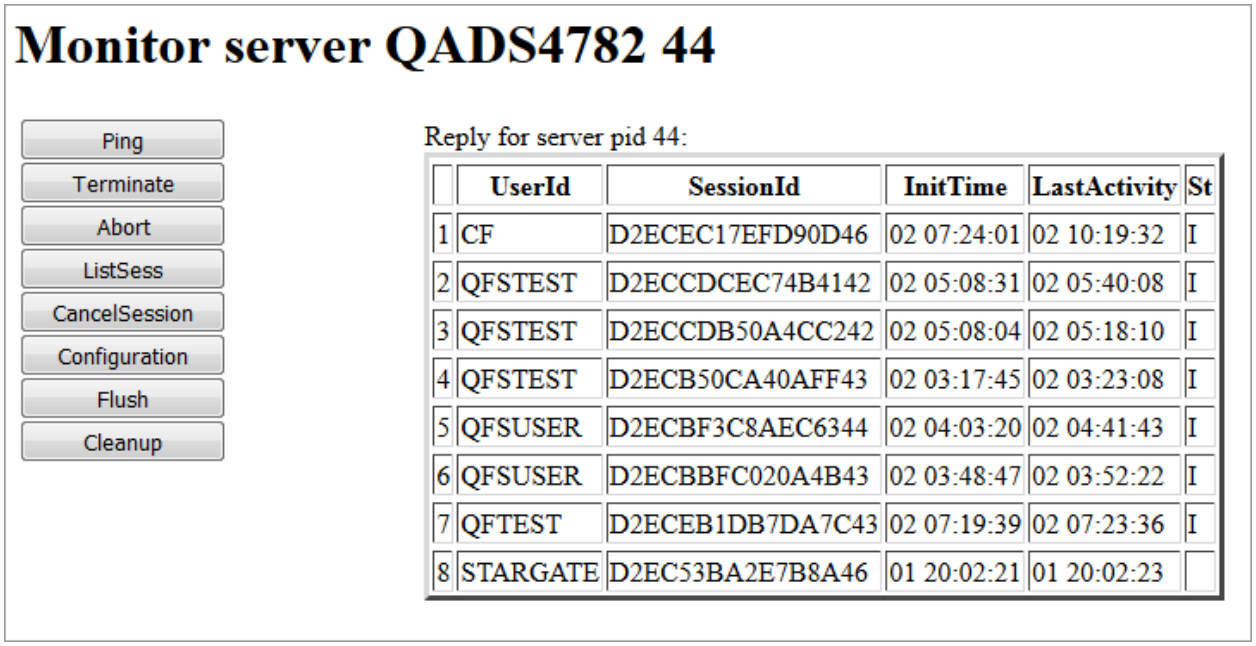
Example:



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.

10 SPoD-Specific Limitations and Considerations

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

- Miscellaneous Restrictions under BS2000

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.

Statements

REQUEST DOCUMENT

The Natural Development Server for BS2000 does not support the REQUEST DOCUMENT statement.

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.

Miscellaneous Restrictions under BS2000

The following restrictions apply when you are using a Natural Development Server (NDV) in a SMARTS server runtime environment on BS2000.

Natural Advanced Facilities

This add-on product cannot be used.

NATRJE

The Natural remote job entry facility `NATRJE` is not currently supported.

Call to External 3GL Subroutines

3GL programs that are to be called by Natural must have the following characteristics:

1. They must be fully reentrant since they are loaded as shared code into a common memory pool.
2. They may not invoke any system function such as request memory or file IO, because they are running in a multitasking environment where every communication with the Natural Studio front-end can cause a task change similar to UTM.
3. AMODE 24 is not supported. Any attempt to load an AMODE 24 subroutine will end up with a GETMAIN error.
4. CRTE is not supported.

The Natural profile parameter `DELETE` is not evaluated; that is, a 3GL program that has been loaded cannot be unloaded.

MT ParameterProfile parameter

The Natural profile and session parameter `MT` (Maximum CPU Time) cannot be used to limit the time consumption.

LIBNAM Parameter

The Natural profile parameter `LIBNAM` (Name of External Program Load Library) is not supported.

Natural System Variables

The Natural system variables `*INIT-PROGRAM`, `*HARDWARE` and `*OSVERS` are not supported.

NDV Server

Only one NDV server can be started per SMARTS server instance.

Terminal Emulation

Currently, only terminal type 3270 is supported.

Natural Shared Nucleus

Support of a Natural shared nucleus is not possible. That is, NDV under SMARTS on BS2000 cannot share one nucleus with Natural under TIAM, UTM, or in batch mode.

Global Bufferpools

The usage of global bufferpools is not possible with NDV under SMARTS on BS2000, because NDV is not able to share bufferpools with Natural under TIAM, UTM, or in batch mode.

Usage of the Software AG Editor under SMARTS on BS2000

Since no editor work file access method is currently available for SMARTS on BS2000, the Software AG editor has to run without a buffer pool.

For this purpose, the profile parameter `EDPSIZE` (described in the Natural *Parameter Reference*) is provided where you can specify the size of an auxiliary editor buffer pool. All editor data are kept in the user storage thread. The total editor work space per user is limited by the `EDPSIZE` parameter. No editor work file is required. The recover mechanism of the Software AG Editor is not supported.

EntireX Broker Stub

The EntireX Broker Stub `BROKER` must be used.

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.

11 Natural Development Server Frequently Asked Questions

This chapter contains frequently asked questions concerning the Natural Development Server (NDV) under SMARTS on BS2000.

Natural Development Server starts and terminates immediately

At server initialization, the Natural Development Server

- allocates central control blocks,
- assigns a central logical system file `SYSOUT` for all tasks,
- 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 or the error output in the logical system file `SYSOUT` 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?

SYSOUT	The central logical system file SYSOUT of the FSIO task contains the error and trace output and the content of the configuration file.

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.

I do not get a NAT0954 even if I specify DU=OFF

SMARTS encountered an irrecoverable error and terminates a worker-task without recovery processing.

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.` 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. **Reason:** ■ Maximum workload of parallel processes exceeded. **Action:** Increase the value of the SMARTS parameters `WORKLOAD_MAXIMUM` and/or `WORKLOAD_AVERAGE`.
3	`Cannot initialize new session.` This error occurs if a storage allocation for internal NDV control buffers fails due to a lack of virtual memory above 16 MB. **Reason:** Virtual memory above 16 MB too small.

Code	Meaning, Reason, Action
	Action: Increase the virtual memory above 16 MB by increasing the value of the SMARTS parameter DATA_MAXIMUM, decrease the number of physical storage threads, distribute the clients to several Natural Development Servers.
4	Session execution failed. Internal error. Natural Studio uses an invalid session identifier to process a request. **Reason:** ■ When a **Map Environment** command is issued, the session ID already exists. ■ The Natural session with the specified ID is not initialized. **Action:** 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. In some situations, a Natural I/O is prohibited at the Natural Development Server. **Reason:** ■ I/O execution during LOGON request. ■ I/O execution while a transaction processor is executing. **Action:** 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. The Natural Development Server cannot finish a terminal I/O. **Reason:** ■ Virtual memory above 16 MB too small. ■ I/O reply buffer sent by Natural Studio is invalid. **Action:** Increase the virtual memory above 16 MB by increasing the value of the SMARTS parameter DATA_MAXIMUM. If the I/O reply buffer is invalid, contact Software AG support.
8	Protocol element missing. Internal error, contact Software AG support.

Code	Meaning, Reason, Action
9	`NDV not installed on Natural system file.` Natural Development Server cannot execute the Natural module `TRPRO` located on library `SYSLIB`. **Reason:** ■ The NDV modules were not loaded on the system file `FNAT`. **Action:** Use the Natural utility INPL to load the NDV modules.
10	`LOGON command required.` 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. 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.

For NDV under BS2000, the translation tables have to be adapted as follows:

ASC2EBCDIC

Position	Value	Character
C4	BB	Ä
D6	BC	Ö
DC	BD	Ü
DF	FF	ß
E4	FB	ä
F6	4F	ö
FC	FD	ü

EBCDIC2ASC

Position	Value	Character
4F	F6	ö
BB	C4	Ä
BC	D6	Ö
BD	DC	Ü
FB	E4	ä
FD	FC	ü
FF	DF	ß

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`.

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.

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.