

Operating the Development Server

This document describes how to operate a Natural Development Server in an OS/390-based batch environment or in a Smarts on VSE/ESA environment.

The following topics are covered:

- Starting the Development Server
- Terminating the Development Server
- Monitoring the Development Server
- Runtime Trace Facility
- Trace Filter

Starting the Development Server

- Under OS/390
- Under Smarts on VSE/ESA

Under OS/390

The development server can be started as a **started task**:

```
//NDVSRV  PROC
//KSPSRV  EXEC PGM=NATRDEVS,REGION=4000K,TIME=1440,
// PARM=( ' POSIX(ON) /NDVSRV1 ' )
//STEPLIB DD DISP=SHR,DSN=NDVvrn.LOAD
//        DD DISP=SHR,DSN=NATvrn.LOAD
//CMPRINT DD SYSOUT=X
//STGCONFIG DD DISP=SHR,DSN=NDV111.CONFIG(SRV1)
//STGTRACE DD SYSOUT=X
//STGSTDO DD SYSOUT=X
//STGSTDE DD SYSOUT=X
```

Where *vrn* is the version, release, system maintenance level number of NDV (starting with NDV111) or Natural (starting with NAT315).

Note: PARM=(' POSIX(ON) /NDVSRV1 ')
 POSIX(ON) is required for a proper LE370 initialization and NDVSRV1 is the name of the server for the communication with the monitor client.

The name of the started task must be defined under RACF and the OS/390 UNIX System Services.

Start the NDV server with the Smarts console command `/F <Smarts-region>,NATRDEVS <server-id>`, where *Smarts-region* is the name of your Smarts started task.

Example: `/F SMARTS62,NATRDEVS NDVS1.`

Note: If you qualify the NDV datasets by the server-id, the server-id is restricted to a maximum length of 7 characters.

Under Smarts on VSE/ESA

Prerequisite is a running Smarts address space that is configured to run NDV (see Development Server Installation under Smarts on VSE/ESA).

Start the NDV server with the Smarts console command `<msg-id> NATRDEVS <server-id>`, where *msg-id* is the message identifier assigned to the Smarts partition and *server-id* is the name of your NDV server.

Example: `141 NATRDEVS NDVS1.`

Note: If you qualify the NDV datasets by the server-id, the server-id is restricted to a maximum length of 6 characters.

Terminating the Development Server

The Development Server can be terminated from within the Monitor Client NATMOPI.

Monitoring the 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 users, terminate the entire server, etc.

Monitor Communication

To communicate with the monitor, you can use the Monitor Client NATMOPI (documented in the Natural for Mainframes Operations documentation).

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 SERVER_TRACE: <code>set SERVER_TRACE 0x00000012</code> |
| 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, last activity time and an internal session identifier (session-id). |
| cancel session session-id | Cancels a particular Natural session within the NDV 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 which can be switched on, if desired.

The following topics are covered:

- Trace Medium
- Trace Configuration
- Trace Level

Trace Medium

A remote development server writes its runtime trace to a dataset allocated to the DD name STGTRACE or, alternatively, to `<server-id>T`.

The trace file is allocated (overwritten) at server initialization.

Trace Configuration

The trace is configured by a trace level which defines the detail of the trace. Once a trace is switched on, it can be restricted to particular clients or client requests by specifying a trace filter.

Every Natural session is equipped with a 32-bit trace status word (TSW) which defines the trace level for that session. The value of the TSW is set by a server configuration parameter. 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 25-24 | 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 13-08 | Free. |
| Bit 07-01 | Free. |
| Bit 00 | Reserved for trace-level extension. |

Trace Filter

In order to reduce the volume of the server trace output, it is possible to restrict the trace by a logical filter.

- The filter can be set with the configuration parameter TRACE_FILTER.
- The filter may consist of multiple keyword=*filtervalue* assignments separated by spaces.

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 and asterisk notation is possible.

Example:

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

Each request of the userid KSP and each request of the userids prefixed by a P are traced.