Operating the Natural Web I/O Interface Server

This chapter describes how to operate a Natural Web I/O Interface server. Unless otherwise noted, the information below applies to all operating systems.

The following topics are covered:

- Starting the Natural Web I/O Interface Server
- Terminating the Natural Web I/O Interface Server under z/OS, z/VSE and VM/CMS
- Terminating the Natural Web I/O Interface Server under BS2000/OSD
- Changing the SYSOUT File Assignment of the FSIO Task under BS2000/OSD
- Monitoring the Natural Web I/O Interface Server
- Runtime Trace Facility
- Trace Filter

Starting the Natural Web I/O Interface Server

Under z/OS:

The Web I/O Interface server can be started as a "started task":

```
//NWOSRV  PROC
//KSPSRV   EXEC PGM=NATRNWO,REGION=4000K,TIME=1440,
// PARM=('POSIX(ON)/NWOSRV1')
//STEPLIB  DD DISP=SHR,DSN=NWOvrs.LOAD
//         DD DISP=SHR,DSN=NATvrs.LOAD
//CMPRINT  DD SYSOUT=X
//STGCONFG DD DISP=SHR,DSN=NWOvrs.CONFIG(SRV1)
//STGTRACE DD SYSOUT=X
//STGSTDO  DD SYSOUT=X
//STGSTDE  DD SYSOUT=X
```

- where

vrs is the version, release, system maintenance level number of NWO or Natural.

Note:
`PARM=('POSIX(ON)/NWOSRV1')` - POSIX(ON) is required for a proper LE370 initialization, and `NWOSRV1` 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 z/OS UNIX System Services.

Uder z/VSE and VM/CMS:

Under z/VSE, a prerequisite is a running SMARTS address space that is configured to run the Natural Web I/O Interface server (see *Installing the Web I/O Server under z/VSE*).

```
<msg-id> NATRNWO <server-id>
```

- where

msg-id is the message identifier assigned to the SMARTS partition, and

server-id is the name of your Natural Web I/O Interface server.

Example for z/VSE:

```
141 NATRNWO NWOS1
```

Under VM/CMS, a prerequisite is a running SMARTS in your CMS that is configured to run the Natural Web I/O Interface server (see *Installing the Web I/O Server under VM/CMS*). If you have a running SMARTS in your CMS, your terminal operates as a SMARTS console where you can enter SMARTS commands.

Under VM/CMS, start the Natural Web I/O Interface server with the SMARTS console command

```
 NATRNWO <server-id>
```

- where

server-id is the name of your Natural Web I/O Interface server.

Example for VM/CMS:

```
NATRNWO NWOS1
```

Note:
If you qualify the Natural Web I/O Interface server datasets by *server-id*, the server ID is restricted to a maximum length of 6 characters.

Alternatively you can automatically start Natural Web I/O Interface servers during SMARTS initialization by using the SMARTS SYSPARM parameter STARTUPPGM. In the SMARTS SYSPARM file specify:

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

Example:

```
STARTUPPGM='NATRNWO NWOS1'
```

Under BS2000/OSD:

Under BS2000/OSD, start the Natural Web I/O Interface server with the SDF command:

```
/ENT-PROCSTART-NWO
```

The SDF procedure START_NWO has to be supplied with the following parameters:

Parameter	Definition	Default Value
NWO-JOBS	The NWO (SMA) job library.	NWO*vrs*.JOBS
ENV-MOD	The NWO environment-specific module library. This library contains the linked Natural nucleus module.	NWO*vrs*.JOBS
NWO-MOD	The NWO module library.	$SAG.NWO*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.	ADA*vrs*.MOD
PROC-NAME	The name of the NWO START procedure. The procedure must reside in the NWO-JOBS library.	START-NWO
NWO-CONFG	The NWO configuration file. It must reside in the NWO-JOBS library (Type S).	NWO-CONFG
NWO-SYSPARM	The SMARTS configuration file. It must reside in the NWO-JOBS library.	NWO-SYSPARM
NWO-ADAPARM	The ADALNK parameter file (IDTNAME, etc). It must reside in the NWO-JOBS library.	NWO-ADAPARM
LOG-FILE-PREFIX	The log-file prefix for the SYSOUT files of all SMARTS tasks.	L.NWO.
WORKER-JOB-NAME	The job name of the worker-tasks.	NWOWORK
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!	

Caution:
Do not modify the variable names and parameter names that are used in the procedure START_NWO. 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.

Example procedure for entering the START-NWO procedure:

```
/ENT-PROC        ($SAG.APSvrs.LIB,START-NWO,J),(                -
/                NWO-JOBS         = $SAG.NWOvrs.JOBS,           -
/                NWO-MOD          = $SAG.NWOvrs.MOD,            -
/                NCF-MOD          = $SAG.NCFvrs.MOD,            -
/                APS-LIB          = $SAG.APSvrs.LIB,            -
/                ADA-MOD          = $SAG.ADAvrs.MOD.NWOvr,      -
/                NWO-SYSPARM      = NWO-SYSPARM,                -
/                NWO-CONFG        = NWO-CONFG,                  -
/                NWO-ADAPARM      = NWO-DDLNKPAR,               -
/                LOG-FILE-PREFIX  = L.NWO.OUT.NWOS01.)
```

Terminating the Natural Web I/O Interface Server under z/OS, z/VSE and VM/CMS

Under z/OS and z/VSE, the Natural Web I/O Interface server can be terminated from within the Monitor Client NATMOPI, see *Monitor Commands* below.

Under VM/CMS, terminate SMARTS with the console command EOJ FORCE.

Terminating the Natural Web I/O Interface Server under BS2000/OSD

The Natural Web I/O Interface server can be terminated with the console command:

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

Changing the SYSOUT File Assignment of the FSIO Task under BS2000/OSD

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:

```
/CLP FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=APSvrs.LIB,ELEMENT=
SHOW-SYSOUT),PROCEDURE-PARAMETERS=(APS-IB=APSvrs.LIB,TSN=7445),
LOGGING=*PARAMETERS"
```

Monitoring the Natural Web I/O Interface Server

To enable the administrator to monitor the status of the Natural Web I/O Interface 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.

Note:
Monitoring is not currently supported under VM/CMS.

The following topics are covered below:

- 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*.

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

Example:

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

sends the command `ping` to the Web I/O Interface server running under the job *jobname*.

Monitor Commands

The Natural Web I/O Interface 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 Web I/O Interface 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.

The following topics are covered below:

- Trace Medium
- Trace Configuration
- Trace Level

Trace Medium

Under z/OS and z/VSE, the Natural Web I/O Interface server writes its runtime trace to the logical system file `SYSOUT` of the `FSIO` task.

Under z/OS, z/VSE and BS2000/OSD, the Natural Web I/O Interface server writes its runtime trace to the logical system file `SYSOUT` of the `FSIO` task.

Under VM/CMS, the Natural Web I/O Interface server writes its runtime trace to a file named `<`*`server id`*`>T` of file type `RTS` to your A disk.

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 Web I/O Interface server 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 Web I/O Interface server 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 send/reply buffer.
Bit 26	Dump send/reply buffer short. Only the first 64 bytes are dumped.
Bit 25	Dump I/O buffer.
Bit 24	Dump I/O buffer short. Only the first 64 bytes are dumped.
Bit 23	Call back gateway event.
Bit 22-16	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

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.