Configuring the Natural Web I/O Interface Server

This chapter describes how to configure a Natural Web I/O Interface server.

Where applicable, specific information for z/OS, z/VSE, VM/CMS or BS2000/OSD is provided.

The following topics are covered:

- Configuration Requirements for z/OS
- Configuration Requirements for z/VSE and VM/CMS
- SMARTS Configuration File for BS2000/OSD
- Web I/O Interface Server Configuration File for z/OS and z/VSE and VM/CMS
- Web I/O Interface Server Configuration Parameters
- Web I/O Interface Server Configuration File Example
- Web I/O Interface Server Datasets for z/OS, z/VSE and VM/CMS
- Web I/O Interface Server User Exits

Configuration Requirements for z/OS

The following topics are covered:

- Language Environment Parameter Settings
- External Security Configuration
- SSL Support

Language Environment Parameter Settings

A Natural Web I/O Interface server requires the following z/OS language environment parameter configuration:

Parameter	Definition	
POSIX(ON)	Enables a Natural Web I/O Interface server to access the POSIX functionality of z/OS. If you start a Natural Web I/O Interface server server with POSIX(OFF), it terminates immediately with a user abend U4093 and the system message EDC5167. IBM supplies the default "OFF".	
TRAP(ON,NOSPIE)	Defines the abend handling of the LE/370 environment:	
	ON	Enables the Language Environment condition handler.
	NOSPIE	Specifies that the Language Environment will handle program interrupts and abends via an ESTAE, that is, the Natural abend handler will receive control to handle program interrupts and abends.
	If you do not specify TRAP(ON,NOSPIE), the Natural abend handling does not work properly. IBM supplies the default (ON,SPIE).	
TERMTHDACT(UADUMP)	Defines the level of information that is produced in case of an abend. The option UADUMP generates a Language Environment CEEDUMP and system dump of the user address space. The CEEDUMP does not contain the Natural relevant storage areas. IBM supplies the default (TRACE).	
ENVAR(TZ=...)	The ENVAR option enables you to set UNIX environment variables. The only environment variable applicable for the Natural Web I/O Interface server is TZ (time zone). This variable allows you to adjust the timestamp within the Natural Web I/O Interface server's trace file to your local time. Example: ENVAR(TZ=CET-1DST) CET - 1 hour daylight saving time	

You can set the z/OS language environment parameters:

- With the PARM parameter specified in the EXEC card of the Natural Web I/O Interface server startup job. The length of the options is limited by the maximum length of the PARM parameter.
- Assemble an LE/370 runtime option module CEEUOPT and link it to the Natural Web I/O Interface server load module.

External Security Configuration

If you configure the Web I/O Interface server to impersonate the Web I/O Interface clients in the server (Web I/O Interface server configuration parameter SECURITY_MODE=IMPERSONATE or IMPERSONATE_LOCAL), the Web I/O Interface server must run "program-controlled". Under RACF, the following definitions are required for the Web I/O Interface server:

- The resource BPX.SERVER must be defined and the Web I/O Interface server account must have READ access to this resource.

- The LOAD datasets defined in the Web I/O Interface server startup job definition must be defined to the program class "**".

```
ralt program ** addmem('natural load library') uacc(read)
ralt program ** addmem('NWO load library'//NOPADCHK) uacc(read)
ralt program ** addmem('user load library'//NOPADCHK) uacc(read)
```

- SETR WHEN(PROGRAM) REFRESH

Additionally, each client connecting to the server must be defined in RACF and must be granted to use the z/OS Unix System Services.

SSL Support

SSL over AT-TLS

SSL support for the Natural Web I/O Interface server is based on the z/OS Communication Server component AT-TLS (Application Transparent-Transport Layer Security).

AT-TLS provides TLS/SSL encryption as a configurable service for sockets applications. It is realized as an additional layer on top of the TCP/IP protocol stack, which exploits the SSL functionality in nearly or even fully transparent mode to sockets applications. AT-TLS offers three modes of operation. See *z/OS Communications Server, IP Programmer's Guide and Reference*. Version 1, Release 9, Chapter 15, IBM manual SC31-8787-09.

These modes are:

- **Basic**

  The sockets application runs without modification in transparent mode, unaware of performing encrypted communication via AT-TLS. Thus legacy applications can run in secured mode without source code modification.

- **Aware**

  The application is aware of running in secured mode and is able to query TLS status information.

- **Controlling**

  The sockets application is aware of AT-TLS and controls the use of AT-TLS encryption services itself. This means, the application is able to switch between secured and non secured communication.

Natural Web I/O Interface server uses the Basic mode for its SSL implementation. That is, a server configured as SSL server rejects requests from non-secured clients.

Maintenance of Certificates under z/OS

Certificates, which are to be used with AT-TLS, can be maintained in two ways under z/OS. They are stored either in RACF key rings or in key databases, which are located in the z/OS UNIX file system. Which of these proceedings actually applies is defined in the AT-TLS Policy Agent Configuration file for the z/OS TCP/IP stack, which is used by the Natural HTTPS client.

IBM delivers a set of commonly used CA root certificates with each z/OS system delivery. If key rings are going to be used to hold server certificates, those root certificates must be manually imported into the key rings by the system administrator. If IBM delivers newer replacements for expired root certificates, all affected key rings have to be updated accordingly.

Unlike key rings, key databases contain the current set of root certificates automatically after they have been newly created. However, the need for maintaining always the latest set of root certificates applies to the key database alternative as well.

Using RACF Key Rings

In RACF, digital certificates are stored in so called key rings. The RACF command `RACDCERT` is used to create and maintain key rings and certificates, which are contained in those key rings.

See *z/OS Security Server RACF Security Administrator's Guide*, IBM manual SA22-7683-11, and *z/OS Security Server RACF Command Language Reference*, IBM manual SA22-7687-11.

Using Key Databases

Alternatively to RACF, certificates can be kept in key databases, which reside in the z/OS UNIX services file system. For the creation and maintenance of key databases, the `GSKKYMAN` utility has to be used.

See *z/OS Cryptographic Services PKI Services Guide and Reference*, IBM manual SA22-7693-10.

How to configure TCP/IP for AT-TLS?

Proceed as follows:

1. In the TCP/IP configuration file, set the option `TTLS` in the `TCPCONFIG` statement.
2. Configure and start the AT-TLS Policy Agent. This agent is called by TCP/IP on each new TCP connection to check if the connection is SSL.
3. Create the Policy Agent file containing the AT-TLS rules. The Policy Agent file contains the rules to stipulate which connection is SSL.

See also *z/OS Communications Server: IP Configuration Guide*, Chapter 18 *Application Transparent Transport Layer Security (AT-TLS) data protection*.

The Sample Policy Agent file defines the server with the job name starting with `NWODEV` and listening at port 4843 to use SSL.

The sample expects the certificate database on the HFS file */u/admin/CERT.kdb*.

```
TTLSRule                          ConnRule01~1
{
  LocalAddrSetRef                 addr1
  RemoteAddrSetRef                addr1
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Jobname                         NWODEV*
  Direction                       Inbound
  Priority                        255
  TTLSGroupActionRef              gAct1~NWO_Server
  TTLSEnvironmentActionRef        eAct1~NWO_Server
  TTLSConnectionActionRef         cAct1~NWO_Server
```

```
}
TTLSGroupAction                   gAct1~NWO_Server
{
  TTLSEnabled                     On
}
TTLSEnvironmentAction             eAct1~NWO_Server
{
  HandshakeRole                   Server
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR1
}
TTLSConnectionAction              cAct1~NWO_Server
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~AT-TLS__Silver
  TTLSConnectionAdvancedParmsRef  cAdv1~NWO_Server
}
TTLSConnectionAdvancedParms       cAdv1~NWO_Server
{
  CertificateLabel                NDV_TEST_CERT
}
TTLSKeyringParms                  keyR1
{
  Keyring                         /u/admin/CERT.kdb
  KeyringStashFile                /u/admin/CERT.sth
}
TTLSCipherParms                   cipher1~AT-TLS__Silver
{
  V3CipherSuites                  TLS_RSA_WITH_DES_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddrSet                         addr1
{
  Prefix                          0.0.0.0/0
}
PortRange                         portR1
{
  Port                            4843
}
PortRange                         portR2
{
  Port                            1024-65535
}
```

How to Verify AT-TLS Configuration?

Check Policy-Agent job output `JESMSGLG` for:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR <your TCP/IP address space>: TTLS
```

This message indicates a successful initialization.

Check Policy-Agent job output `JESMSGLG` for:

```
EZZ8438I PAGENT POLICY DEFINITIONS CONTAIN ERRORS FOR <your TCP/IP address space>: TTLS
```

This message indicates errors in the configuration file. Check the *syslog.log* file for further information.

Does the configuration rule cover the server?

Try to connect the server and check *syslog.log* for:

```
EZD1281I TTLS Map   CONNID: 00018BED LOCAL: 10.20.91.61..4843 REMOTE: 10.20.160.47..4889 JOBNAME: NWODEV42 USERID: NWOSRV TYPE: InBound STATUS: Enabled RULE: ConnRule01~1 ACTIONS: gAct1 eAct1~NWO_Server cAct1~NWO_Server
```

The above entry indicates that the connection to Port 4843 is SSL enabled.

Frequently Asked Questions

Is there more information about problem determination?

See also *z/OS V1R8.0 Comm Svr: IP Diagnosis Guide: 3.23*, Chapter 29 *Diagnosing Application Transparent Transport Layer Security (AT-TLS)*

How to switch on P-agent trace?

See *Comm Svr: IP Configuration Reference*, Chapter 20 *Syslog deamon and Comm Svr: IP Configuration Guide*, Chapter 1.5.1 *Configuring the syslog daemon (syslogd)*

Error at connection establishment

Find return code RC and corresponding `GSK_` function name in P-agent trace.

See *System SSL Programming* and locate the RC in Chapter 12.1 *SSL Function Return Codes*.

Sample trace with `trace=255`:

```
EZD1281I TTLS Map   CONNID: 00002909 LOCAL: 10.20.91.61..1751 REMOTE: 10.20.91.117..443 JOBNAME: KSP USERID: KSP TYPE: OutBound STATUS: A
EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000000 CONNID: 00002909  RC:    0 Connection Init
EZD1282I TTLS Start GRPID: 00000003 ENVID: 00000002 CONNID: 00002909 Initial Handshake ACTIONS: gAct1 eAct1 AllUsersAsClient HS-Client
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Call GSK_SECURE_SOCKET_OPEN - 7EE4F718
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_SESSION_TYPE -  CLIENT
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_V3_CIPHER_SPECS -  090A2F
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_FD - 00002909
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_USER_DATA - 7EEE9B50
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:  435 Call GSK_SECURE_SOCKET_INIT - 7EE4F718
EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:  435 Initial Handshake 00000000 7EEE8118
EZD1286I TTLS Error GRPID: 00000003 ENVID: 00000002 CONNID: 00002909 JOBNAME: KSP USERID: KSP RULE: ConnRule01  RC:  435 Initial Handshake
EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Connection Close 00000000 7EEE8118
```

Generation of a Natural Web I/O Interface Server Certificate

Under z/OS, SSL certificates can be produced with the UNIX System Services utility `GSKKYMAN`. The following steps have to be executed for the production of a new certificate, which is to be used for the SSL secured communication between Natural Web I/O Interface client and server:

1. Start a shell session out of TSO or connect via telnet to the z/OS UNIX shell.
2. Start `GSKKYMAN`.
3. Create a new – or open an existing key database.
4. Create a self-signed certficate, for example, of type "User or server certificate with 2048-bit RSA key".
5. Export the certificate to a (HFS) file. Choose Base64 ASN.1 DER as export format.

This generated file can now be copied to the Natural Web I/O Interface client(s) using FTP with ASCII transfer format. On the client side, the received file should be stored with the file name suffix *.CER*. The certificate can now be used by the Natural Web I/O Interface client.

If certificates are kept in a RACF key ring, the generated certificate has to be imported into the appropriate key ring using the `RADCERT` command.

Certificates, which are produced on a different platform, for example, on a Windows PC, can be imported into a RACF key ring or into a key database as well.

Detailed information about the use of the `GSKKYMAN` utility can be found in the IBM Communications server documentation, e.g in the following manuals:

z/OS Communications Server IP Configuration Guide Version 1 Release 2 (IBM manual SC31-8775-01

or

z/OS Communications Server Cryptographic Services System Secure Sockets Layer Programming (IBM manual SC24-5901-04).

For the generation of certificates under Windows, a free downloadable utility named Ikeyman is available on several websites. Ikeyman is an IBM product as well and maps the functionality of `GSKKYMAN` to the Windows platform.

Configuration Requirements for z/VSE and VM/CMS

The following topics are covered below:

- SMARTS SYSPARM Parameters
- SYSPARM Example for the Natural Web I/O Interface Server

SMARTS SYSPARM Parameters

The Natural Web I/O Interface server requires the following SMARTS `SYSPARM` parameters:

Parameter	Definition
`RESIDENTPAGE`	The following members must be defined in the SMARTS resident area: `NATRNWO`, `NATMONI`, `NATDSSEC`, Natural front-end (`NCFNUC`) and Natural nucleus (if you run using a split nucleus).
`SECSYS`	The installed external security system (`RACF` | `ACF2` | `TOPSECRET`).

Parameter	Definition	
SERVER	The following SERVER definitions are required for the Natural Web I/O Interface server:	
	SERVER=(OPERATOR,TLINOPER,TLSPOPER)	The Operator Communications Server.
	SERVER=(POSIX,PAENKERN)	The POSIX Server.
	The Natural local buffer pool definition.	For details, refer to the Natural Com-plete interface documentation.
CDI_DRIVER	CDI_DRIVER=('TCPIP,PAACSOCK,MINQ=10,MAXQ=20')	The SMARTS TCP/IP Socket Driver for Connectivity Systems TCP/IP stack on z/VSE and VM/CMS. MINQ/MAXQ define the number of TcpIP listener tasks.

Parameter	Definition	
THSIZEABOVE	`THSIZEABOVE=1024`	The storage above 16 MB that is available for each Natural Web I/O Interface server subtask. This size must be large enough to keep the Natural tread, heap and stack of the Natural Web I/O Interface server subtasks. A certain headroom (20% or more, depending on your environment) is recommended. If the Natural Web I/O Interface server initialization fails with `NAT9915 GETMAIN for thread storage failed`, this parameter must be increased.
ADASVC	`ADASVC=`*`nnn`*	The Adabas SVC number of your Adabas installation.

You can set the SMARTS `SYSPARM` parameters in the file `SMARTS.CONFIG` which must reside on one of your accessed disk.

SYSPARM Example for the Natural Web I/O Interface Server

For z/VSE:

```
* ------------------ ADABAS PARMS -----------------------------*
ADACALLS=20 CALLS BEFORE ROLL
ADASVC=47 ADABAS SVC NUMBER
* ------------------ BUFFERPOOL PARMS -------------------------*
BUFFERPOOL=(064,030,20,ANY)
BUFFERPOOL=(128,064,64,ANY)
BUFFERPOOL=(256,010,10,ANY)
BUFFERPOOL=(512,032,10,ANY)
BUFFERPOOL=(1K,032,32,ANY)
BUFFERPOOL=(6K,005,02,ANY)
BUFFERPOOL=(8K,016,16,ANY)
* ------------------ ROLLING PARMS ----------------------------*
ROLL-BUFFERPOOL=(048K,04,04,DS) ESA DATA SPACE
ROLL-BUFFERPOOL=(064K,04,04,DS) ESA DATA SPACE
ROLL-BUFFERPOOL=(128K,04,04,DS) ESA DATA SPACE
ROLL-BUFFERPOOL=(256K,04,04,DS) ESA DATA SPACE
ROLL-BUFFERPOOL=(800K,02,02,DS) ESA DATA SPACE
*
* ------------------ NWO Server to launch at startup ----------*
* STARTUPPGM=’NATRNWO NWOS1’
*

*
TASK-GROUP=(DEFAULT,6)
THREAD-GROUP=(DEFAULT,(DEFAULT,252,06,15,28,N))
*
THSIZEABOVE=1024
*
SERVER=(NCFNAT42,NCFNAT42)   NAT42 BUFFER POOL
*
CDI_DRIVER=(’TCPIP,PAACSOCK,MINQ=10,MAXQ=20’)
*
RESIDENTPAGE=NATRNWO
RESIDENTPAGE=NWONCF42
RESIDENTPAGE=NATNUC42
RESIDENTPAGE=NATMONI
```

For VM/CMS:

```
*
*        SMARTS/SSE System Parameters
*
WORKLOAD-AVERAGE=10
*
WORKLOAD-MAXIMUM=50
*
TIBTAB=DYN00040
*
* TASK-GROUP=(DEFAULT,2)
*THREAD-GROUP=(DEFAULT,(DEFAULT,252,02,15,28,N))
*
*
* Recovery
*ABEND_RECOVERY=NO                        MAIN TASK RECOVERY YES/NO
*THREAD_ABEND_RECOVERY=NO                 THREAD RECOVERY YES/NO
CDI_DRIVER=(TCPIP,PAASSOCK)
ADASVC=247
*----------------- SERVERS ------------------------------------------*
CDI_DRIVER=(PIPE,PAANPIO)
CDI_DRIVER=(CONSOLE,PAANCNIO)
*
SERVER=(OPERATOR,TLINOPER)            OPERATOR COMMAND READER
```

```
*
SERVER=(POSIX,PAENKERN)               POSIX SERVER DEFINITION
SERVER=(NCFNAT41,NCFNAT41)
STARTUPPGM='NATRNWO NWO21'
RESIDENTPAGE=NATRNWO
RESIDENTPAGE=NATMONI
RESIDENTPAGE=NATMOPI
RESIDENTPAGE=NATHTMON
RESIDENTPAGE=NCF422
RESIDENTPAGE=NAT422RE
RESIDENTPAGE=NCFNAT42
RESIDENTPAGE=NCFBTIO
RESIDENTPAGE=NCFWFAPS
*
THSIZEABOVE=2048              THREAD AREA > 16 M ADDRESS
* ENVIRONMENT_VARIABLES=DD:CONFIG
CDI_DRIVER=(FILE,PAASFSIO)
SYSTEM_TRACE_LEVEL=5
* * * End of File * * *
```

SMARTS Configuration File for BS2000/OSD

The Natural Web I/O Interface server reads its configuration parameters from a file which resides as an S-type member in the NWO-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 Web I/O Interface Server under BS2000/OSD* in the Natural for Windows (Natural Studio) documentation.
2. The SMARTS configuration is contained as an S-type member which resides in the NWO-JOBLIB. It has to be passed to the system in the startup procedure parameter NWO-SYSPARM.

The following topics are covered below:

- SMARTS SYSPARM Parameters
- SMARTS Server Environment Configuration Parameters
- SMARTS Sample Configuration

SMARTS SYSPARM Parameters

The Natural Web I/O Interface 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: NATRNWO, NATMONI, Natural front-end (NCFNUC) and Natural nucleus.

Parameter	Definition	
SERVER	The following SERVER definitions are required for the Natural Web I/O Interface server:	
	SERVER=(POSIX,PAENKERN,CONF=PAANCONF)	The POSIX server.
	SERVER=(NCFNAT*nn*,NCFNAT*nn*)	The Natural buffer pool server. The size of the various Natural buffer pools is configured in the parameter string of the NWO configuration parameter SESSION_PARAMETER.
	SERVER=(NWOSERV,PAENAPPS,NATRNWO,'*SERVERID*')	The NWO 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 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 (e.g. 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 *Using SMARTS PFS under BS2000/OSD*.
MOUNT_FS	MOUNT_FS=('*PFS*://','/')	Mapping of (PFS) file names to the appropriate physical BS2000/OSD container file; see also *Using SMARTS PFS under BS2000/OSD*. This parameter maps (POSIX) filenames to a physical BS2000/OSD container file for the specified PFS.

Parameter	Definition	
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/OSD 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, NWO 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 value defines the number of sessions that can be active and kept in storage in parallel; see also the WORKLOAD and TASK-GROUP parameters.

Parameter	Definition	
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/OSD tasks) that can execute NWO 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 the parameter module NCFPARM). A certain headroom (20% or more, depending on your environment) is recommended. If the Natural Web I/O Interface server initialization fails with NAT9915 GETMAIN for thread storage failed, this parameter must be increased.

Parameter	Definition	
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 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 Web I/O Interface server under SMARTS on BS2000/OSD.

- 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 NWO-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/OSD.
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 kilobytes and 1 megabyte.
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 `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-GROUP`s, 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=34
THSIZEABOVE=4096
THREAD-GROUP=(DEFAULT,(DEFAULT,0,4)
TASK-GROUP=(DEFAULT,2)
WORKLOAD-AVERAGE=8
WORKLOAD-MAXIMUM=40
RESIDENTPAGE=NATSOCK
RESIDENTPAGE=NATMONI
```

```
RESIDENTPAGE=NATRNWO
RESIDENTPAGE=NCFSERV
RESIDENTPAGE=NvrLPRRB
*********************SERVERS ****************************
SERVER=(POSIX,PAENKERN,CONF=PAANCONF)
SERVER=(NCFNATvr,NCFNATvr,1,2048,2,100,4,2048)
SERVER=(NWONATv,PAENAPPS,NATRNWO,'NWOS01')
*********************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:NWO/ROOT/SERVER1/')
MOUNT_FS=('PFS://','/')
```

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

Web I/O Interface Server Configuration File for z/OS and z/VSE and VM/CMS

A configuration file is allocated to the name <*serverid*>C (for example, NWOS1C) or STGCONFG alternatively.

The configuration file is a text file located on a dataset or on an HFS file under z/OS and a librarian member under z/VSE and a librarian member under VM/CMS.

Note:
Under VM/CMS, the configuration file is a text file of type RTS located on one of your accessed disks.

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 *Web I/O Interface Server Configuration File Example* shown below.

Web I/O Interface Server Configuration Parameters

The following Web I/O Interface server configuration parameters are available:

- COMPATIBILITY_MODE
- FRONTEND_NAME
- FRONTEND_OPTIONS
- FRONTEND_PARAMETER
- HANDLE_ABEND
- HOST_NAME
- HTPMON_ADMIN_PSW

- HTPMON_PORT
- HOST_NAME
- IGNORE_PRESENT_SERVER
- INITIAL_USERID
- KEEP_TCB
- PASSWORD_MIXEDCASE
- PORT_NUMBER
- SECURITY_MODE
- SESSION_PARAMETER
- THREAD_NUMBER
- THREAD_SIZE
- TRACE_FILTER
- TRACE_LEVEL
- UPPERCASE_SYSTEMMESSAGES

COMPATIBILITY_MODE

The current version of NWO presumes to run with the most recent version of Natural. An error `NAT7729 NWO and Natural version do not agree` is issued when running with older Natural versions. This is because NWO must negotiate a subset of functionality with the client at a time when the involved Natural version is not already known.

If you want to run NWO with a previous version of Natural, you can set this parameter to `YES`. It is recommended that you leave this parameter at its default value if you intend to run your NWO with the most recent version of Natural, because in this case `COMPATIBILITY_MODE=YES` would unnecessarily limit the functionality.

Value	Explanation
`YES`	Accept also older versions of Natural. Results in a limitation of the functionality documented with the most recent version.
`NO`	Presume to run with the most recent version of Natural. This is the default value.

Example:

```
COMPATIBILITY_MODE=YES
```

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
```

FRONTEND_OPTIONS

This configuration parameter applies to z/OS only.

The values of this configuration parameter may be used to specify additional options for the Natural front-end.

Value	Explanation
01	Do not use the Roll Server. This is the default value.
02	Clean up roll file at server termination.
04	Write GTF trace.
08	Write ETRACE.
10	Front-end automatic termination.
20	Write console information.

You may combine the above options as desired in that you add their values and set the result as shown in the example below.

Example:

```
FRONTEND_OPTIONS=07
```

The setting in this example enables the Options 01, 02 and 04.

FRONTEND_PARAMETER

This configuration parameter applies to z/OS only.

This optional configuration parameter contains additional Natural front-end parameters as specified in the Startup Parameter Area.

Value	Explanation
parameter-name	You can define multiple parameters. Each parameter specification is a pair of 8-character strings, the first containing the parameter keyword and the second the parameter value, for example: `FRONTEND PARAMETER = 'MSGCLASSX  '`

No default value is provided.

For further information, refer to the section *Natural in Batch Mode* in the Natural *Operations for Mainframe* documentation.

Example:

```
FRONTEND_PARAMETER='MSGCLASSX  '
```

The setting in this example specifies that the default output class for `CMPRINT` is "X".

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 NWO server has its own abend recovery.

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

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

Example:

```
HANDLE_ABEND=NO
```

HOST_NAME

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

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

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

No default value is provided.

Example:

```
HOST_NAME=node1
```

or

```
HOST_NAME=157.189.160.55
```

HTPMON_ADMIN_PSW

This configuration parameter applies to z/OS, z/VSE and BS2000/OSD.

This configuration parameter defines the password required for some monitor activities (e.g. 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

This configuration parameter applies to z/OS, z/VSE and BS2000/OSD.

A Web I/O Interface 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 applies to z/VSE, VM/CMS and BS2000/OSD.

This configuration parameter defines the host name of the Web I/O Interface server.

IGNORE_PRESENT_SERVER

This configuration parameter applies to z/OS and z/VSE in conjunction with the Web I/O Interface server CICS Adapter.

A Web I/O Interface (NWO) server allocates a so-called "server environment" which contains the server dependent common resources.

This environment is unique for each NWO server and relates to the server name. If an NWO server with Web I/O Interface Server CICS Adapter ends abnormally, it might leave a stuck NWO server environment within the CICS region. This causes that a restart of the server fails with error message NAT9913.

If you start an NWO server with `IGNORE_PRESENT_SERVER=YES`, it might damage an already running server which is using the same server name and the same CICS region.

Value	Explanation
`YES`	Terminate existing CICS server environment.
`NO`	Abort server initialization if a CICS server environment already exist. This is the default value.

Example:

```
IGNORE_PRESENT_SERVER=YES
```

INITIAL_USERID

At server initialization, the Natural Web I/O Interface 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=NWOINITU
```

See also *Web I/O Interface clients must be defined to Natural Security* in the operating-system-specific Natural Web I/O Interface server *Installation* section.

KEEP_TCB

This configuration parameter applies to z/OS only.

By default, the remote Natural session of a mapped environment terminates its TCB whenever you switch the focus within Natural Studio to a different mapped environment. If you toggle the focus back, the remote session is dispatched using a different TCB.

The maximum number of active TCBs is equal to the number of connected clients.

The configuration parameter `KEEP_TCB` specifies whether the remote Natural session should use the same TCB during its entire lifetime. This is required if you want to access DB2. It could also be required if you access 3GL programs which need to be executed under the same TCB for successive calls.

Value	Explanation
`YES`	The remote Natural session uses the same TCB during its entire lifetime.
`NO`	This is the default value.

Example:

```
KEEP_TCB=YES
```

PASSWORD_MIXEDCASE

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

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

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

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

Example:

```
PASSWORD_MIXEDCASE=YES
```

PORT_NUMBER

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

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

No default value is provided.

Example:

```
PORT_NUMBER=3140
```

Note:
Under BS2000/OSD, some port numbers are privileged and reserved for certain system services. Ask your BS2000/OSD system administrator for the port number range available to you.

SECURITY_MODE

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

The Natural Web I/O Interface server offers a security concept that also covers the operating system resources. The client credentials are validated at the operating-system-depending security system and the client request is executed under the client's account data.

Using the `SECURITY_MODE` parameter, you can specify at which rank (in batch mode or under CICS) you want to impersonate the activities of a Web I/O Interface client.

Value	Explanation
`IMPERSONATE_LOCAL`	Impersonation is done within the Natural Web I/O Interface server environment. If the session is dispatched in a remote TP environment (e.g. in CICS using the NWO CICS Adapter), it is still executed anonymous. The client must be defined in the security system of the Web I/O Interface server. It is not required to define the client in a remote TP environment. For z/OS, see also *External Security Configuration*. For z/VSE and VM/CMS, see also *SYSPARM Parameter SECSYS.*
`IMPERSONATE_REMOTE`	No impersonation is done within the Natural Web I/O Interface server environment. If the session is dispatched in a remote TP environment, the client is impersonated. The client must be defined in the security system of the remote TP environment. See also Web I/O Interface server security exit NATUXRFE and the section *Product Interaction* in the *Web I/O Interface Server CICS Adapter* documentation. **Note:** Please verify the correct installation of `NATUXRFE`. A **Map Environment** attempt with a valid user ID and an invalid password should fail with a NAT0873 error.
`IMPERSONATE`	Impersonation is done within the Natural Web I/O Interface server environment and in a remote TP environment. The client must be defined in the security system of the Natural Web I/O Interface server and in the remote TP environment.

No default value is provided.

Example:

```
SECURITY_MODE=IMPERSONATE
```

SESSION_PARAMETER

This optional configuration parameter defines session parameters that precede the parameter string specified in the connection dialog of the Natural Web I/O Interface client.

Value	Explanation
parameter-string	This string may extend across several lines. A + 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=(NWOPARM,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 Web I/O Interface 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`.

THREAD_NUMBER

This configuration parameter applies to z/OS only.

This configuration parameter specifies the number of physical storage threads to be allocated by the Natural front-end, that is, the number of sessions that can be executed in parallel.

Note:
This parameter is obsolete when the Natural Web I/O Interface Server CICS Adapter is used.

Value	Explanation
thread-number	Number of physical storage threads to be allocated. **Note:** This number does not limit the number of sessions within the server, but the number of sessions which can be in execution status concurrently. The number of sessions is limited by the size of the Natural swap medium.
3	This is the default value.

Example:

```
THREAD_NUMBER=5
```

THREAD_SIZE

This configuration parameter applies to z/OS only.

This configuration parameter specifies the size of each physical storage thread which contains the Natural session data at execution time.

Note:
This parameter is obsolete when the Natural Web I/O Interface Server CICS Adapter is used.

Value	Explanation
thread-size	Size (in KB) of each physical storage thread.
500	This is the default value.

Example:

```
THREAD_SIZE=800
```

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 Web I/O Interface Server*.

TRACE_LEVEL

Value	Explanation
trace-level	See *Trace Level* in the section *Operating the Natural Web I/O Interface 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.

UPPERCASE_SYSTEMMESSAGES

This configuration parameter is used to enable or disable the translation of all NWO 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.

Web I/O Interface Server Configuration File Example

For z/OS:

```
# This is a comment
SESSION_PARAMETER=profile=(stgqa,10,930) fuser=(10,32) CFICU=ON
THREAD_NUMBER=2
THREAD_SIZE=700
FRONTEND_NAME=NATOS42L       # and another comment
PORT_NUMBER=4811
```

For z/VSE:

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

For VM/CMS:

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

For BS2000/OSD:

```
SESSION_PARAMETER = 'NUCNAME=N42LPRRB' +
  ' PROFILE=(NWOPARM,18006,58),ADAMODE=0,' +
  'FNAT=(18006,58),FUSER=(18006,19),FDIC=(18006,11),' +
  'FSPOOL=(18006,58),FSEC=(18006,58),CFICU=ON,MENU=OFF,CP=EDF03IRV'
THREAD_NUMBER = 3
THREAD_SIZE = 900
FRONTEND_OPTIONS = 0X01
FRONTEND_NAME = NCFSERV
PORT_NUMBER = 4811
MONITOR=Y
```

Web I/O Interface Server Datasets for z/OS, z/VSE and VM/CMS

The Natural Web I/O Interface server requires the following datasets:

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

Alternately, you can qualify each dataset name by the server ID. Under z/VSE and VM/CMS, this is necessary if you want to start different Natural Web I/O Interface servers under a single SMARTS address space.

NWOS1C	Defines the server configuration file for the server NWOS1.
NWOS1T	The server trace output for the server NWOS1.
NWOS1O	The stdo dataset for the server NWOS1.
NWOS1E	The stde error output for the server NWOS1.

Web I/O Interface Server User Exits

The Natural Web I/O Interface server offers the following user exit:

User Exit NSECUX01

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

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

If the exit (the loadmodule `NSECUX01`) is found in the NWO load library concatenation, it is called before the user is validated against RACF.

The following parameters are passed to the exit:

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

The exit is called using standard linkage conventions.

Sample user exit implemented in C:

```
#include <string.h>
#include <stdio.h>

#  pragma linkage (NSECUX01, FETCHABLE)

void NSECUX01(char sUid[64])
{
char sUidTemp[64];

  printf("Uex got usid:%s\n", sUid);
  strcpy(sUidTemp, sUid);
  sprintf(sUid, "$$%s", sUidTemp);
  printf("Uex ret usid:%s\n", sUid);
  return;
}
```

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