Natural for Mainframes

Natural Web I/O Interface

Version 9.2.2

June 2025

ADABAS & NATURAL

Dieses Dokument gilt für Natural for Mainframes ab Version 9.2.2.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Dokument-ID: NATMF-NNATWEBIO-922-20250606

Table of Contents

Preface

This documentation is organized under the following headings:

Introduction	What is the Natural Web I/O Interface?
Introducing the Natural Web I/O Interface Server CICS Adapter	Special information which applies if you want to use the Natural Web I/O Interface server in a CICS environment under z/OS.
Introducing the Natural Web I/O Interface Server IMS Adapter	Special information which applies if you want to use the Natural Web I/O Interface server in an IMS TM environment under z/OS.
Installing and Configuring the Natural Web I/O Interface Server	How to install and configure the Natural Web I/O Interface server in a mainframe environment.
Installing the Natural Web I/O Interface Client	How to install the Natural Web I/O Interface client on an application server or in a servlet container so that it can be used with the Natural Web I/O Interface server.
Configuring and Administering Clients	How to define the information that is to appear in the logon page.
Operating and Monitoring the Natural Web I/O Interface Server	How to operate the Natural Web I/O Interface server in a mainframe environment, and how to monitor it using the Monitor Client `NATMOPI` or the HTML Monitor Client.

Note: This documentation only explains how to install the Natural Web I/O Interface server in a mainframe environment. For information on how to install it in a Linux or Windows environment, see the Natural documentation for the appropriate platform.

1 About this Documentation

Dokumentationskonventionen

Konvention	Beschreibung
Fettschrift	>Kennzeichnet Elemente auf einem Bildschirm.
`Nichtproportionale Schrift`	Kennzeichnet Namen und Orte von Diensten im Format *`Ordner.Unterordner.Dienst`*, Programmierschnittstellen (APIs), Namen von Klassen, Methoden und Properties in Java.
`Kursivschrift`	Kennzeichnet: Variablen, für die Sie situations- oder umgebungsspezifische Werte angeben müssen. Neue Begriffe, wenn sie erstmals im Text auftreten. Verweise auf andere Dokumentationsquellen.
`Nichtproportionale Schrift`	Kennzeichnet: Text, den Sie eingeben müssen. Meldungen, die vom System angezeigt werden. Programmcode.
{ }	Zeigt eine Reihe von Auswahlmöglichkeiten an, von denen Sie eine auswählen müssen. Geben Sie nur die innerhalb der geschweiften Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole { } ein.
|	Trennt zwei sich gegenseitig ausschließende Auswahlmöglichkeiten in einer Syntaxzeile voneinander ab. Geben Sie eine der Auswahlmöglichkeiten ein. Geben Sie nicht das Symbol | ein.
[ ]	Zeigt eine oder mehrere Optionen an. Geben Sie nur die innerhalb der eckigen Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole [ ] ein.
...	Zeigt an, dass Sie mehrere Auswahlmöglichkeiten desselben Typs eingeben können. Geben Sie nur die Informationen ein. Geben Sie nicht die drei Auslassungspunkte (...) ein.

Online-Informationen und Support

Produktdokumentation

Sie finden die Produktdokumentation auf unserer Dokumentationswebsite unter **https://documentation.softwareag.com**.

Zusätzlich können Sie auch über **https://www.softwareag.cloud** auf die Dokumentation für die Cloud-Produkte zugreifen. Navigieren Sie zum gewünschten Produkt und gehen Sie dann, je nach Produkt, zu „Developer Center“, „User Center“ oder „Documentation“.

Produktschulungen

Sie finden hilfreiches Produktschulungsmaterial auf unserem Lernportal unter **https://knowledge.softwareag.com**.

Tech Community

Auf der Website unserer Tech Community unter **https://techcommunity.softwareag.com** können Sie mit Experten der Software AG zusammenarbeiten. Von hier aus können Sie zum Beispiel:

- Unsere umfangreiche Wissensdatenbank durchsuchen.
- In unseren Diskussionsforen Fragen stellen und Antworten finden.
- Die neuesten Nachrichten und Ankündigungen der Software AG lesen.
- Unsere Communities erkunden.
- Unsere öffentlichen Repositories auf GitHub and Docker unter **https://github.com/softwareag** und **https://hub.docker.com/publishers/softwareag** besuchen und weitere Ressourcen der Software AG entdecken.

Produktsupport

Support für die Produkte der Software AG steht lizenzierten Kunden über unser Empower-Portal unter **https://empower.softwareag.com** zur Verfügung. Für viele Dienstleistungen auf diesem Portal benötigen Sie ein Konto. Wenn Sie noch keines haben, dann können Sie es unter **https://empower.softwareag.com/register** beantragen. Sobald Sie ein Konto haben, können Sie zum Beispiel:

- Produkte, Aktualisierungen und Programmkorrekturen herunterladen.
- Das Knowledge Center nach technischen Informationen und Tipps durchsuchen.
- Frühwarnungen und kritische Alarme abonnieren.
- Supportfälle öffnen und aktualisieren.
- Anfragen für neue Produktmerkmale einreichen.

Datenschutz

Die Produkte der Software AG stellen Funktionen zur Verarbeitung von personenbezogenen Daten gemäß der Datenschutz-Grundverordnung (DSGVO) der Europäischen Union zur Verfügung. Gegebenenfalls sind in der betreffenden Systemverwaltungsdokumentation entsprechende Schritte dokumentiert.

I Introduction

2 Introduction

This chapter describes the purpose and the functions of the Natural Web I/O Interface.

Note: This introduction mainly describes how the Natural Web I/O Interface works in a runtime (production) environment. The section *Differences in a SPoD Development Environment* briefly explains the special version that is used in a SPoD development environment.

What is the Natural Web I/O Interface?

The Natural Web I/O Interface is used to execute Natural applications in a web browser. It fully supports the following:

- The display and input of Unicode characters. See *Unicode Input/Output Handling in Natural Applications* in the *Unicode and Code Page Support* documentation.
- Rich internet applications developed with Natural for Ajax.

Components of the Natural Web I/O Interface

The Natural Web I/O Interface consists of a server and a client.

Server

The Natural Web I/O Interface server enables you to use a browser as the I/O device for Natural applications. The server does the user authentication, creates the Natural session and handles the I/O between Natural and the client. The Natural Web I/O Interface server is installed on the same machine as the Natural application.

Client

The client handles the communication between the user's web browser and the Natural Web I/O Interface server. It converts the output from the Natural application to web pages, and returns the user input to Natural.

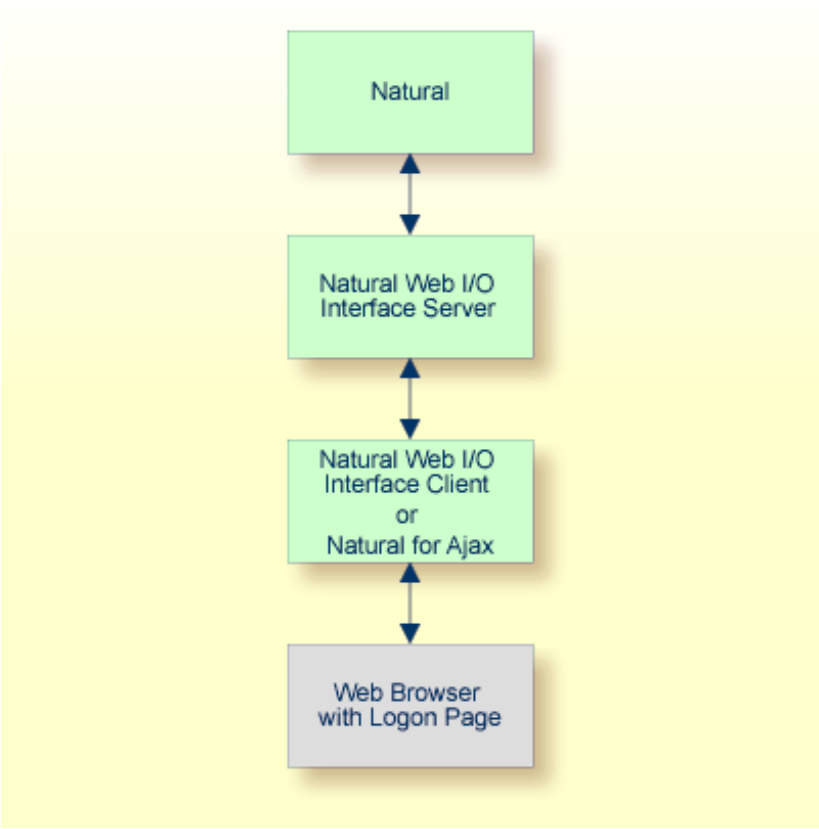
Two types of client are supported:

- Natural Web I/O Interface client for displaying character-based applications in the web browser. Maps with GUI controls are not supported in this case.
- Natural for Ajax for displaying rich internet applications in the web browser. For further information on this type of client, see the Natural for Ajax documentation.

The client is installed on a web/application server. This can be done on any machine in the network.

Executing a Natural Application in a Web Browser

The Natural Web I/O Interface receives data from a Natural application and delivers web pages to the user's web browser. This is illustrated in the following graphic:



The communication steps for executing a Natural application in the web browser are:

1. The user enters the address (URL) of a logon page in the web browser. The client then displays the logon page in the web browser.

   Note: For information on how to invoke and configure the logon page, see *Configuring the Client*.

2. The user enters all required information for starting a Natural application into the logon page. This information is sent to the client.
3. The client asks the Natural Web I/O Interface server to start the requested Natural application for this user.
4. The Natural Web I/O Interface server checks the supplied user ID and password, creates a Natural session for the user and starts the Natural application.

5. The Natural application returns the first application screen which is then transferred via the Natural Web I/O Interface server to the client and finally as a web page to the web browser.

Different web browsers are supported. Note that cookies and JavaScript must be enabled in the web browser. For a list of the currently supported web browsers, see the browser prerequisites for the type of client that you are using.

Client-Server Compatibility

The following rules apply:

- The Natural Web I/O Interface server can work with any client that has the same or a higher protocol version.

  If the server detects that the client is using a version that is lower than the server version, the server replies that the client is too old and the connection is closed.
- The client can work with any server that has the same or a lower protocol version.

  If the client detects that the server is using a version that is lower than the client version, the client switches to the server version. However, new client functionality is not supported in this case.
- The Natural Web I/O Interface server must have the same protocol version as the Natural process that is started by the server. If Natural detects that the server is using a different protocol version, an error message is sent to the user and the connection is closed.

Terminology

On the different Natural platforms for which the Natural Web I/O Interface is supported, different techiques are used for implementing the server part of the Natural Web I/O Interface. On Natural for Linux, it is implemented as a daemon. On Natural for Windows, it is implemented as a service. On the mainframe, it is implemented as a server. In this documentation, the general term “server” is therefore used for all different kinds of implementation.

Differences in a SPoD Development Environment

The previous sections of this introduction have described how the Natural Web I/O Interface works in a runtime (production) environment. This section briefly explains the differences in a SPoD development environment.

A special version of the Natural Web I/O Interface is used when working in a remote development environment with Natural for Windows (SPoD). In this case, the Natural Web I/O Interface is an integrated component which does not require a separate installation. The server is part of the Natural Development Server (NDV), and the client is part of Natural Studio. Other than in the runtime environment, the screen is not displayed in a browser but in a normal window. Rich GUI pages created by Natural for Ajax are not supported in the development environment.

It is important that I/O via the Natural Web I/O Interface has been enabled on the Natural host. Otherwise, the Natural Web I/O Interface cannot be invoked. See also *Unicode Input/Output Handling in Natural Applications* in the *Unicode and Code Page Support* documentation.

Restrictions When Using the Natural Web I/O Interface with Natural Applications

There are several restrictions when using the Natural Web I/O Interface with Natural applications on Linux, mainframe or Windows hosts.

Note: The term “application” refers to application software. It does not refer to system software or software for development.

The following restrictions apply:

- **GUI controls**
  GUI controls are not supported: dialogs, buttons, radio buttons, list boxes, list views, check boxes etc. The Natural Web I/O Interface only supports Natural applications developed without GUI controls.
- **File transfer**
  File transfer (for example, with the `DOWNLOAD` statement) is not supported by the Natural Web I/O Interface.
- **Runtime errors**
  This restriction applies to older Natural versions on Linux and Windows. As of version 6.3.3, this restriction no longer applies.

Runtime errors in Natural applications are not handled by the Natural Web I/O Interface. This leads to a loss of the session. Bypass: use the Natural system variable *ERROR-TA to handle the error. Sample Natural error transaction:

```
DEFINE DATA
LOCAL
1 ERR_INFO
  2 ERR_NR(N5)
  2 ERR_LINE(N4)
  2 ERR_STAT(A1)
  2 ERR_PNAM(A8)
  2 ERR_LEVEL(N2)
END-DEFINE
INPUT ERR_INFO
DISPLAY ERR_INFO
TERMINATE
END
```

- **Terminal commands**
  Terminal commands are not supported. They do not work when entered in the Natural Web I/O Interface client.
- **Natural system variable** *INIT-ID
  When using the Natural Web I/O Interface client with Natural applications on Linux, mainframe or Windows hosts, the Natural system variable *INIT-ID will not be filled with a value for the terminal type. On Linux and Windows, it will contain the value "notty". On mainframes, it will contain a session ID that is unique on that server.

The following restrictions apply to Natural on Linux and Windows hosts (the mainframe does not have these restrictions):

- **Return to the Natural main screen**
  You must not use Natural applications that return to the Natural main screen as this leads to wrong screen display and a loss of the session.
- **Natural editors and utilities**
  You must not use Natural utilities such as SYSMAIN or SYSDDM and editors such as the program editor as this leads to wrong screen display and a loss of the session.
- **Natural system commands**
  You must not use any Natural system command such as CATALL, FIND, GLOBALS, HELP, KEY, LIST, RETURN, SCAN, SETUP or XREF as this leads to wrong screen display and a loss of the session.

Differences between the Natural Web I/O Interface Client and Terminal Emulation

The Natural Web I/O Interface client runs as an HTML terminal emulator inside a browser control. The look and feel of the Natural Web I/O Interface client display is quite similar to that of the regular terminal (emulation), but there are some differences due to browser functionality:

- A double-click with the mouse pointer on any field simulates the ENTER key.
- It is not possible to position the cursor outside the range of input and output fields.
- The cursor can be moved with the left and right arrow keys within one input field. It is also possible to jump from one input field to another using the left, right, up and down arrow keys.
- The insert mode can be switched on and off using the INSERT key.
- For Unicode character sets (type U; for example, Chinese), one character may require more space than an ordinary alphanumeric character, because the Unicode character representation is proportional. The application design must take this into account, because Natural is based on characters with fixed width. For input fields it is possible to scroll within the field, but for output fields there may not be sufficient space to display the Unicode characters. The display length for a field can be controlled by the session parameter DL.
- Type-ahead mode is not supported.
- Paste in overwrite mode is not supported.
- Key schemes are fixed; keys such as the right CTRL key and the ENTER key on the numeric pad are no longer definable.
- Screen update is slower since the complete screen is sent rather than updates.
- The blink attribute is not supported in Internet Explorer.
- The keys PF1 through PF12 are simulated by the key combinations F1 through F12.
- The keys PF13 through PF24 are simulated by the key combinations SHIFT+F1 through SHIFT+F12.
- The keys PF25 through PF36 are simulated by the key combinations CTRL+F1 through CTRL+F12.
- The keys PF37 through PF48 are simulated by the key combinations ALT+F1 through ALT+F12.
- The program attention keys (PA1, PA2 and PA3) are simulated by the key combinations CTRL+SHIFT+F1, CTRL+SHIFT+F2, CTRL+SHIFT+F3.
- The clear key is simulated by CTRL+SHIFT+F4.

IBM Mainframes Only

- The terminal screen size is controlled by the Natural profile parameter TMODEL. The default setting TMODEL=0 means 24 lines and 80 columns.
- There is no ATTN (attention interrupt) key, no RESET key and no EEOF (erase end of file) key.

VT Only

The I/O occurs in block mode. Therefore, the Natural program will only react when a function key is pressed.

II Introducing the Natural Web I/O Interface Server CICS Adapter

3 Introducing the Natural Web I/O Interface Server CICS Adapter

This chapter describes the purpose and the functions of the Natural Web I/O Interface Server CICS Adapter.

Purpose of the Natural Web I/O Interface Server CICS Adapter

The Natural Web I/O Interface Server CICS Adapter is designed for a mainframe Natural context where it enables the use of a Natural Web I/O Interface server, running under z/OS in batch mode within a CICS TP monitor environment.

CICS Support

The CICS support is not implemented within the front-end stub `NATRNWO`. For dispatching the Natural sessions in CICS, the Web I/O Interface server continues to run in batch mode or under SMARTS. But it uses the remote front-end `NATCSRFE` that is delivered with the Natural Web I/O Interface server to dispatch the Natural sessions in CICS. That is, depending on the installed front-end, a server dispatches the sessions locally (`NCFNUC` for SMARTS, `NATMVS` for batch mode) or remotely (`NATCSRFE` for CICS).

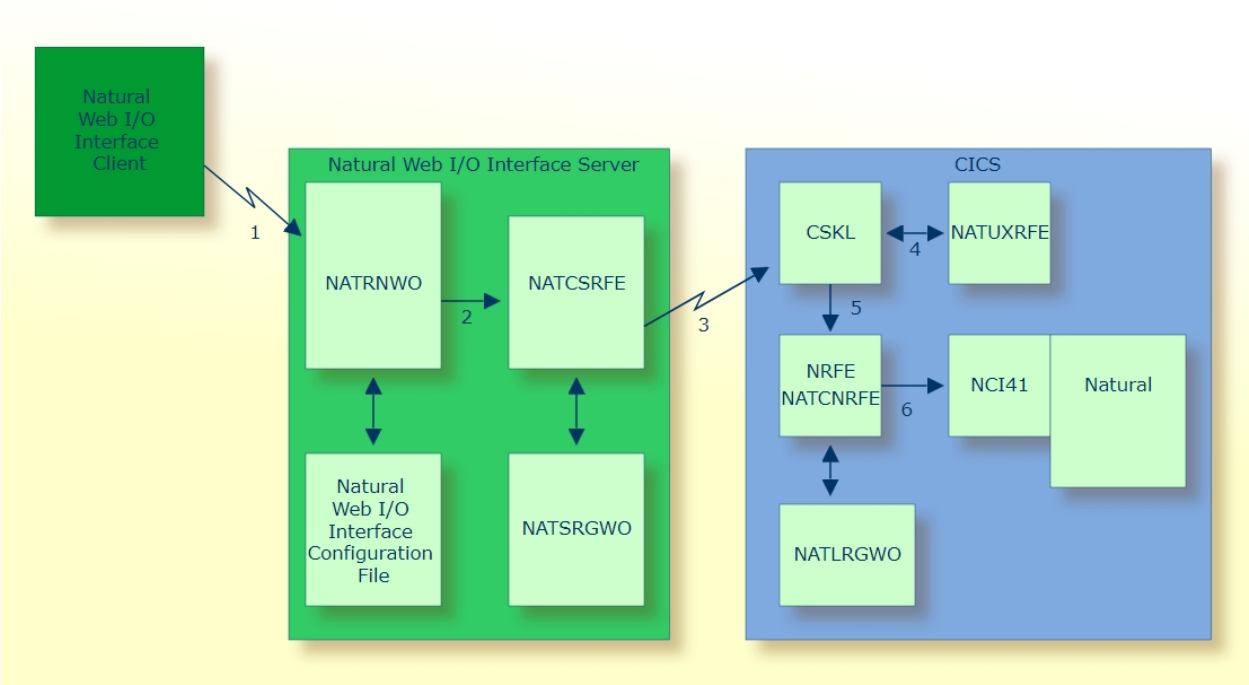
`NATCSRFE` in turn accepts the Natural request from `NATRNWO` and transfers it to a configured CICS environment using the CICS Socket Interface. Within the CICS environment, a CICS Natural transaction is launched that processes the Natural request and returns the result. Thus it is not necessary to execute the entire Web I/O Interface server under CICS. Only if Natural is requested to run the Natural application, control is transferred to CICS for execution.

The Natural Web I/O Interface Server CICS Adapter comprises the following components:

`NATCSRFE`	The remote front-end called by the Natural Web I/O Interface server to dispatch a Natural request. It is loaded into the Web I/O Interface server's address space.
`NATCNRFE`	The counterpart of `NATCSRFE`. `NATCNRFE` runs in the CICS address space. It is started by the IBM-provided standard listener of the CICS Socket Interface (refer also to *TCP/IP V3R1 for MVS: CICS TCP/IP Socket Interface Guide*).
`NATSRGWO/NATLRGWO`	Transmits the data relevant for Natural Web I/O Interface server between Natural Web I/O Interface server and the Natural session running in CICS. `NATSRGWO` must be loaded into the Natural Web I/O Interface server's address space and `NATLRGWO` into the CICS address space.
`NATUXRFE`	This user exit obtains the client credentials from the Natural Web I/O Interface server and authenticates then with a CICS `VERIFY PASSWORD` request. If the request succeeds, the CICS listener launches the NWO transaction under the client account (impersonation).

Product Interaction

The following figure illustrates the interaction between the Natural Web I/O Interface server and the CICS environment involved.



1. The Web I/O Interface (NWO) client sends a request to the Natural Web I/O Interface server using the port number specified with the Natural Web I/O Interface server configuration variable `PORT_NUMBER`.
2. The Natural Web I/O Interface server dispatches the Natural session using the Natural front-end you have specified with the Natural Web I/O Interface server configuration variable `FRONTEND_NAME`. Specify `NATCSRFE` in order to use the Natural Web I/O Interface Server CICS Adapter.
3. NATCSRFE transmits the request to the host/port specified with the Natural Web I/O Interface server configuration variable `RFE_CICS_TA_HOST` / `RFE_CICS_TA_PORT`. You must configure the CICS-supplied standard listener `CSKL` to listen at this port.
4. If the Natural Web I/O Interface server is configured to perform remote impersonation (`SECURITY_MODE=IMPERSONATE/IMPERSONATE_REMOTE`), `NATUXRFE` is called to authenticate the client. If the authentication succeeds, `CSKL` launches the CICS transaction `NRFE` under the account of the client (impersonated).
5. `CSKL` launches the CICS transaction you have specified with the Natural Web I/O Interface server configuration parameter `RFE_CICS_TA_NAME` (`NRFE` in this example). This transaction must be defined to use the program `NATCNRFE`.

6. NATCNRFE finally dispatches the Natural session using the Natural CICS front-end you have specified with the Natural Web I/O Interface server configuration parameter RFE_CICS_FE_NAME.

III Introducing the Natural Web I/O Interface Server IMS Adapter

4 Introducing the Natural Web I/O Interface Server IMS Adapter

This chapter describes the purpose and the functions of the Natural Web I/O Interface Server IMS Adapter.

Purpose of the Natural Web I/O Interface Server IMS Adapter

The Natural Web I/O Interface Server IMS Adapter is designed for a mainframe Natural context where it enables the use of a Natural Web I/O Interface server running under z/OS in batch mode within an IMS TM environment.

IMS TM Support

The IMS TM support is not implemented within the front-end stub NATRNWO. For dispatching the Natural sessions in IMS TM, the Web I/O Interface server continues to run in batch mode. But it uses the remote front-end NATISRFE that is delivered with the Natural Web I/O Interface server to dispatch the Natural sessions in IMS TM. That is, depending on the installed front-end, a server dispatches the sessions locally (NATMVS for batch mode) or remotely (NATISRFE for IMS TM).

NATISRFE in turn accepts the Natural request from NATRNWO and transfers it to a configured IMS TM environment using the IMS installation provided BPM listener. The IMS listener launches in a dedicated MPP region a non conversational Natural transaction that processes the Natural request and returns the result. Thus it is not necessary to execute the entire Natural Web I/O Interface server under IMS TM. Only small working units (Natural requests such as “save source” or “get library list”) are transferred to IMS TM for execution.

The Natural Web I/O Interface Server IMS Adapter comprises the following components:

NATISRFE	The remote front-end called by the Natural Web I/O Interface server to dispatch a Natural request. It is loaded into the Web I/O Interface server's address space.
NATINRFE	The counterpart of NATISRFE. NATINRFE runs in an MPP region. It is launched by the IBM-provided IMS Listener (refer also to z/OS Communications Server IP IMS Socket Guide).
NATSRGWO/NATLRGWO	Transmits the data relevant for Natural Web I/O Interface server between Natural Web I/O Interface server and the Natural session running in IMS TM. NATSRGWO must be loaded into the Natural Web I/O Interface server's address space, and NATLRGWO into the MPP region.

Product Interaction

The following description explains the interaction between the Natural Web I/O Interface server and the IMS TM environment involved.

1. The Web I/O Interface (NWO) client sends a request to the Natural Web I/O Interface server using the port number specified with the Natural Web I/O Interface server configuration variable PORT_NUMBER.
2. The Natural Web I/O Interface server dispatches the Natural session using the Natural front-end you have specified with the Natural Web I/O Interface server configuration variable FRONTEND_NAME. Specify NATISRFE in order to use the Natural Web I/O Interface Server IMS Adapter.
3. NATISRFE transmits the request to the host/port specified with the Natural Web I/O Interface server configuration variable RFE_IMS_TA_HOST / RFE_IMS_TA_PORT. You must configure the IBM provided BMP Listener to listen at this port.
4. The BMP Listener launches the IMS TM transaction you have specified with the Natural Web I/O Interface server configuration parameter RFE_IMS_TA_NAME. This transaction must be specified in the configuration of the IMS Listener.
5. The server transaction first retrieves the transaction initialization message. TIM contains the necessary information to do the TAKESOCKET and passes it to NATINRFE. NATINRFE does the TAKESOCKET and establishes the TCP/IP conversation with NATISRFE.
6. NATINRFE finally dispatches the Natural IMS front-end, which establishes a Natural server session.

IV Installing and Configuring the Natural Web I/O Interface Server

The Natural Web I/O Interface server is available on z/OS.

Note: 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.

This part covers the following topics:

Natural Web I/O Interface Server Concept and Structure

Prerequisites

Installing the Natural Web I/O Interface Server under z/OS

Configuring the Natural Web I/O Interface Server

Natural Web I/O Interface Server CICS Adapter
The following topics apply in addition if you want to use the Natural Web I/O Interface server in a CICS environment under z/OS:

Installing the Natural Web I/O Interface Server CICS Adapter under z/OS

Configuring the Natural Web I/O Interface Server CICS Adapter

Natural Web I/O Interface Server IMS Adapter
The following topics apply in addition if you want to use the Natural Web I/O Interface server in an IMS TM environment under z/OS:

Installing the Natural Web I/O Interface Server IMS Adapter

Configuring the Natural Web I/O Interface Server IMS Adapter

Notation *vrs* or *vr*

When used in this documentation, the notation *vrs* or *vr* represents the relevant product version (see also *Version* in the *Glossary*).

5 Natural Web I/O Interface Server Concept and Structure

This chapter describes the concept and the structure of the server for the Natural Web I/O Interface which is designed for use on z/OS.

Web I/O Interface Server Concept

A Natural Web I/O Interface server is a multi-user, multi-tasking application. It can host Natural sessions for multiple users and execute their applications concurrently.

The concept is based on the "serverized" Natural runtime system. Its architecture comprises a server front-end stub (Web I/O Interface server stub `NATRNWO`) that uses the Natural front-end to dispatch Natural sessions and to execute applications within these sessions.

The Natural Web I/O Interface server architecture basically consists of:

- **Front-end stub**

  The stub `NATRNWO` is launched to initialize a Natural Web I/O Interface server. It listens for incoming connection requests an launches a Natural session for executing the application.

- **Front-end**

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

- **Server monitor**

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

Front-End Stub NATRNWO

The multi-user, multi-tasking, front-end stub `NATRNWO` is launched to initialize a Natural Web I/O Interface server.

The following topics are covered below:

Stub Description

The task executing the server initialization (`TMain`) basically is the main listener which waits for incoming requests from the Web I/O Interface client. 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 server connection issued by the user on the client side (the Login button of the Web I/O Interface client) 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` initializes a new Natural session and starts the specified Natural application program.
- While the application performs I/O requests, `TWork` intercepts the I/O data and passes them to the Web I/O Interface client for processing the I/O. The I/O reply is sent back to the server and the server continues the application.
- If the application terminates (reaches the `NEXT` mode), TWork terminates the Natural session and drops the connection to the Web I/O Interface client.

That is, each client owns one subtask `TWork` on the Natural Web I/O Interface server. This subtask runs a Natural session (and within the Natural session, a Natural application) and remains active as long as the application is running.

Natural System Variables Used

Within a Natural Web I/O Interface server session, the following Natural system variables are used:

- `*TPSYS` contains `SERVSTUB`,
- `*DEVICE` contains `BROWSER`,
- `*SERVER-TYPE` contains `WEBIO`.

Front-End

Under z/OS, the Natural front-end required for a Natural Web I/O Interface server is a Natural batch front-end driver, which should be LE enabled. See sample installation jobs for details.

Under z/OS, 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.

The Natural front-end required for executing the Natural sessions under control of CICS is the Natural remote front-end NATCSRFE that is delivered with the Natural Web I/O Interface server. For more information, refer to the *Natural Web I/O Interface Server CICS Adapter* documentation.

Server Monitor

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, the administrator can control the server activities, cancel particular user sessions, terminate the entire server, etc. See ***Operating the Web I/O Interface Server***.

6 Prerequisites

This chapter describes the prerequisites that apply when you install a Natural Web I/O Interface server on a mainframe computer.

General Prerequisites for Web I/O Interface Server Installation

The following general prerequisites apply:

- The currently applicable version of Natural for Mainframes must be installed.
- The International Components for Unicode for Software AG (ICS) must be installed with the Web I/O Interface server; see *International Components for Unicode for Software AG* in the *Unicode and Code Page Support* documentation.

Note: For further information, refer to the products and versions specified under *Software AG Product Versions Supported by Natural* in the current *Natural Release Notes*.

Prerequisites for the Web I/O Interface Server for z/OS

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

- z/OS must be installed.
- To prevent the formation of endless loops in user programs running under the Web I/O Interface, specify a reasonable value for Natural profile parameter `MT` (maximum CPU time).

7 Installing the Natural Web I/O Interface Server under z/OS

This chapter describes how to install a server for the Natural Web I/O Interface (product code NWO) under the operating system z/OS.

The installation of the Web I/O Interface server is performed by installation jobs. The sample jobs are contained in the data set NWO*vrs*.JOBS (where *vrs* represents the relevant product version) and are prefixed with NWO, or generated by System Maintenance Aid (SMA).

Notation *vrs* or *vr*

When used in this documentation, the notation *vrs* or *vr* represents the relevant product version (see also *Version* in the *Glossary*).

Prerequisites

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

Content of the Web I/O Interface Server Distribution Medium

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

Data Set Name	Contents
NWO*vrs*.OBJS	Contains the object modules of the server.
NWO*vrs*.JOBS	Example installation jobs.

Installation Procedure

Step 1: Allocate the Web I/O Interface Server LOAD library

(Job I008, Step 9410)

Step 2: Create a Web I/O Interface Server configuration file and sample Clist

(Job I009 / Step 9410, 9420, 9430)

Step 9410 creates a sample `NWOCONFG` for the batch server.

Step 9420 creates a sample `Clist` to ping and terminate a Web I/O Interface server.

Step 9430 creates a sample batch job to ping and terminate a Web I/O Interface server.

The following parameters of the configuration file have to be defined. See *Configuring the Natural Web I/O Interface Server*. For the other parameters, the default values may be used:

`FRONTEND_NAME`	Specify the name of the Web I/O Interface server front-end module you will generate in one of the following steps.
`PORT_NUMBER`	Specify the TCP/IP port number under which the server can be connected.

Step 3: Link the object modules into the NWO load library

(Job I054, Step 9410)

The NWO object modules must be linked with the necessary runtime extensions of your batch installations into executable load modules.

Note: The module `NATCNRFE` applies to two different products, the Natural Web I/O Interface Server (NWO) and the Natural Development Server (NDV). So if you have already installed NDV, the module `NATCNRFE` might already be there. However, it does not matter if you re-install `NATCNRFE` with NWO because the resulting module from either installation is the same.

See sample job `NWOI054` on data set `NWO`*`vrs`*`.JOBS`.

Step 4: Create the Natural parameter module and NWO server front-end module

(Job I060, Steps 9410, 9420, 9430)

- Job I060, Step 9410 starts the batch program `IEBUPDATE` to store the parameter module `NWOPARM`.
- Job I060, Step 9420 assembles and links the parameter module `NWOPARM`.
- Job I060, Step 9430 links the NWO server front-end module.

  The reentrant `ADALINK` module `ADALNKR` must be used.

  Note: If you have Natural subproducts for IBM database access installed, for example, Natural for Db2, Natural for VSAM, you need to adjust the link job to include `ATRRCSS` from `SYS1.CSSLIB` in order to enable the use of `SYNCPOINT/ROLLBACK` functionalities.

Step 5: Create server startup JCL

(Job I200, Step 9415)

Described in the section *Configuring the Natural Web I/O Interface Server*. See sample member NWOSTART on data set NWO*vrs*.JOBS.

Step 9415 creates a startup procedure for the batch server.

Sample:

```
//          PROC SRV=SAGNWO
//NWO       EXEC PGM=NATRNWO,
// REGION=4000K,TIME=1440,PARM='POSIX(ON),TRAP(ON,NOSPIE)/&SRV'
//STEPLIB   DD   DISP=SHR,DSN=NWOvrs.LOAD
//          DD   DISP=SHR,DSN=SAGLIB.SMALOAD
//SYSUDUMP  DD   SYSOUT=X
//CEEDUMP   DD   SYSOUT=X
//CMPRINT   DD   SYSOUT=X
//STGCONFG  DD   DISP=SHR,
//               DSN=NWO.CONFIG(&SRV)
//STGTRACE  DD   SYSOUT=X
//STGSTDO   DD   SYSOUT=X
//STGSTDE   DD   SYSOUT=X
//SYSOUT    DD   SYSOUT=X
```

Note: The Web I/O Interface server account must be defined in the z/OS Linux System Services (OE segment). If the server account is not defined, the server ends with U4093 and system message CEE5101C in the trace file.

Step 6: Web I/O Interface clients must be defined to Natural Security

If Natural Security (NSC) is installed:

- The Web I/O Interface initial user ID (default ID is STARGATE) must be defined in Natural Security with a valid default library. Refer also to Web I/O Interface server configuration parameter INITIAL_USERID. Alternatively, you can define the Natural profile parameter AUTO=OFF (automatic logon) for the Web I/O Interface.
- Each client user ID must be defined in Natural Security.

If the Web I/O Interface initial user ID is not defined, the Web I/O Interface server initialization aborts with a NAT0856.

If a Web I/O Interface client is not defined, the server connection returns an NSC error.

If you connect to the server from a Web I/O Interface client, make sure that the user who is defined in Natural Security has a default library or a private library defined. Otherwise, error message NAT0815 will occur.

Step 7: Web I/O Interface clients must be defined to the server host

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

8 Configuring the Natural Web I/O Interface Server

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

Configuration Requirements for z/OS

The following topics are covered:

- Language Environment Parameter Settings
- External Security Configuration
- SSL/TLS 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 Linux 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:	

Parameter	Definition
	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 data sets 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 Linux System Services.

SSL/TLS Support

- SSL/TLS over AT-TLS
- Maintenance of Certificates under z/OS
- Using RACF Key Rings
- Using Key Databases
- How to configure TCP/IP for AT-TLS?
- How to Verify AT-TLS Configuration?
- Frequently Asked Questions

- Generation of a Natural Web I/O Interface Server Certificate

SSL/TLS over AT-TLS

SSL/TLS 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 SSL/TLS 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/TLS 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*. Chapter: *Application Transparent Transport Layer Security (AT-TLS)*.

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/TLS implementation. That is, a server configured as SSL/TLS 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 Linux 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*, Chapter: *RACF and digital certificates - RACF and key rings*, and *z/OS Security Server RACF, Command Language Reference*, Chapter: *RACF command syntax - RACDCERT*.

Using Key Databases

Alternatively to RACF, certificates can be kept in key databases, which reside in the z/OS Linux 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*, Chapter: *Appendix B. Using a gskkyman key database for your certificate store*.

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/TLS.
3. Create the Policy Agent file containing the AT-TLS rules. The Policy Agent file contains the rules to stipulate which connection is SSL/TLS.

See also *z/OS Communications Server: IP Configuration Guide*, Chapter: *Application Transparent Transport Layer Security 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/TLS.

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: NWODEVvr 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/TLS enabled.

Frequently Asked Questions

Is there more information about problem determination?

See z/*OS Communications Server, IP Diagnosis Guide*, Chapter: *Diagnosing Application Transparent Transport Layer Security (AT-TLS)*

How to switch on P-agent trace?

See z/*OS Communications Server, IP Configuration Reference*, Chapter: *Syslog deamon*, and z/*OS Communications Server, IP Configuration Guide*, Chapter: *Base TCP/IP system – TCP/IP Customization - Configuring the syslog daemon (syslogd)*.

Error at connection establishment

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

Get description of the return code from *z/OS Cryptographic Services, System Security Sockets Layer Programming*, Chapter: *: messages and codes – 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/TLS certificates can be produced with the Linux 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/TLS 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 Linux 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

or

z/OS Cryptographic Services, System Secure Sockets Layer Programming

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.

Web I/O Interface Server Configuration File for z/OS

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 data set or on an HFS file under z/OS.

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

See also the ***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
- FORCE_IPV4
- FRONTEND_NAME
- FRONTEND_OPTIONS
- FRONTEND_PARAMETER
- HANDLE_ABEND
- HOST_NAME
- HTPMON_ADMIN_PSW

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
```

FORCE_IPV4

This parameter is only available with Natural Web I/O Interface Version 8.3 or above.

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

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

FRONTEND_NAME

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

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

No default value is provided.

Example:

```
FRONTEND_NAME=NAT<vrs>SV
```

FRONTEND_OPTIONS

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

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

No default value is provided.

Example:

```
HOST_NAME=node1
```

or

```
HOST_NAME=157.189.160.55
```

HTPMON_ADMIN_PSW

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

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

IGNORE_PRESENT_SERVER

This configuration parameter applies 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

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 use Adabas and the Adabas parameter ADANAME is set to ADAUSER or 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
```

O4I

Note: This parameter is only available with Natural Web I/O Interface Version 8.3 or above.

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.

Example:

```
O4I=YES
```

PASSWORD_MIXEDCASE

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
```

SECURITY_MODE

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 z/OS 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. See also *External Security Configuration*.
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.

Note: For a batch server, SECURITY_MODE=IMPERSONATE and SECURITY_MODE=IMPERSONATE_LOCAL are the same. SECURITY_MODE requires Natural Security (NSC).

No default value is provided.

Example:

```
SECURITY_MODE=IMPERSONATE
```

SECURITY_SYSLOG

Write a console message in case of security problems with external security. One line is written to the console, for example '`Aug-23 16:22:46.77 S0305217 BPXM023I (INITUSER) SERVERID - Invalid userid specified.`' More information about the reason can be found in the appropriate trace or error log of the server.

The parameter applies to z/OS Batch servers only.

Value	Explanation
YES	Write security message to console.
NO	Do not write security message to console. This is the default value.

Example:

```
SECURITY_SYSLOG=YES
```

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)'
```

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`.

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.

THREAD_NUMBER

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 or Natural Web I/O Interface Server IMS 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 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 or Natural Web I/O Interface Server IMS 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

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

Where *vr* is the current product version and release number.

Web I/O Interface Server Data Sets for z/OS

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

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

Alternately, you can qualify each data set name by the server ID.

NWOS1C	Defines the server configuration file for the server NWOS1.
NWOS1T	The server trace output for the server NWOS1.
NWOS1O	The stdo data set 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 load module NSECUX01) is found in the NWO load library concatenation, it is called using standard linkage conventions (direct branch using a BASR instruction) 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.

9 Installing the Natural Web I/O Interface Server CICS Adapter under z/OS

This chapter describes how to install the CICS connection for a Natural Web I/O Interface server (product code NWO) running under z/OS in batch mode.

Prerequisites

For details, refer to the section *Prerequisites*.

Installation Procedure

To install the Natural Web I/O Interface Server CICS Adapter, perform the following steps:

Step 1: Customize CICS

(Job I005, Steps 9405, 9406, 9410, 9411)

The Natural Web I/O Interface server load library must be defined in the CICS `DFHRPL` concatenation.

The CICS TCP/IP environment can be customized using the configuration macro `EZACICD` in the `EZACONFG` configuration data set or the configuration transaction `EZAC`. This section describes the usage of the transaction `EZAC`.

Customize the standard listener `CSKL` of the CICS socket interface using the CICS transaction `EZAC,ALTER,LISTENER` and, on the second screen, define `NATUXRFE` in the `SECEXIT`.

For impersonation with pass phrases an enhanced listener is required. Customize an existing enhanced listener or define a new one using the CICS transaction `EZAC,DEFINE,LISTENER` with `TRANID CSEL` (or another name of your choice) and `FORMAT ENHANCED`. Customize this enhanced listener by altering the settings on the second screen as follows:

```
CSTRANid     ===> NRFE
CSSTTYPe     ===> KC
CSDELAY      ===> 000000
MSGLENgth    ===> 000
PEEKDATa     ===> NO
MSGFORMat    ===> EBCDIC
USEREXIT     ===> NATUXRFE
GETTID       ===> NO
USERID       ===>
```

The definition of `SECEXIT=NATUXRFE` is mandatory when an enhanced listener is used or when the Natural Web I/O Interface server is started with impersonation (parameter `SECURITY_MODE`).

Start the listener using the CICS transaction `EZAO`.

Specify AF=INET6 if IPv6 (Internet Protocol Version 6) is to be used.

Note: Using IPv6 is only possible with Natural Web I/O Interface Server Version 8.3.2 or above

The following CICS resource definitions are required:

1. Define the CICS transaction for the remote front-end. This transaction name is an arbitrary name which must be defined in the NWO configuration parameter RFE_CICS_TA_NAME. This document uses the transaction name NRFE.

```
DEFINE TRANSACTION(NRFE) GROUP(NWOgroup)
PROGRAM(NATCNRFE)
TWASIZE(128)
RESTART(NO)
TASKDATAKEY(USER)
TASKDATALOC(ANY)
```

2. Define the programs NATCNRFE, NATLRGWO and NATUXRFE:

```
DEFINE PROGRAM(NATCNRFE) GROUP(NWOgroup)
LANGUAGE(C) DATALOCATION(ANY) EXECKEY(USER)
```

```
DEFINE PROGRAM(NATLRGWO) GROUP(NWOgroup)
LANGUAGE(C) DATALOCATION(ANY) EXECKEY(USER)
```

```
DEFINE PROGRAM(NATUXRFE) GROUP(NWOgroup)
LANGUAGE(LE370) DATALOCATION(ANY) EXECKEY(CICS)
```

NATUXRFE must be defined with EXECKEY(CICS) because the transaction CSKL is defined with TASKDATAKEY(CICS) and the program NATUXRFE is part of the calling chain initiated by CSKL. This also applies if another transaction defined with TASKDATAKEY(CICS) is used to invoke NATUXRFE.

In addition, when using the CICS open transaction environment (OTE), set the following parameters for NATCNRFE and NATUXRFE:

```
API(OPENAPI)
CONCURRENCY(THREADSAFE)
```

This is required, for example, if the parameter OTE=YES is set for the configuration macro EZACICD with TYPE=CICS for the CICS region, or if the parameter is set with the transaction EZAC,ALTER,CICS.

The values of API, CONCURRENCY and EXECKEY for NATCNRFE must be the same as for the environment-dependent nucleus of Natural because Natural is called by NATCNRFE using standard

linkage conventions (direct branch using a BASR instruction) instead of an `EXEC CICS LINK` command.

You need not adapt the program definition of `NATLRGWO` for the CICS OTE.

3. For DB2 access, a DB2 plan name must be defined. If you have not specified a DB2 plan name for pool threads in the `DB2CONN` resource definition, the transaction specified in `RFE_CICS_TA_NAME` and its associated DB2 plan name must be defined to CICS with a `DB2TRAN` and/or `DB2ENTRY` resource definition.

   Note: The dynamic plan selection provided by the Natural for DB2 interface must not be used.

4. For DB2 access, the authorization ID under which the NWO CICS transaction is accessing DB2 must have the necessary privileges for DB2 access. The authorization ID to be used is specified in the `DB2ENTRY` resource definition. If you choose the `USERID` option, the user ID of the CICS system will be used because the NWO CICS transactions are running under the user ID of the CICS system.

The sample JCL containing the following members defines all necessary CICS entries:

- NWOCONF

Step 2: Create member for CICS server

- Job I009, Step 9411, create member `NWOCONFC` for CICS server

Step 3: Link the object modules into the NWO load library

(Job I054, Step 8420)

The NWO object modules must be linked with the necessary runtime extensions of your CICS installations into executable load modules.

See sample job `NWOI054` on data set `NWO`*`vrs`*`.JOBS`.

Step 4: Create startup procedure

- Job I200, Step 9416, create startup procedure for CICS server

Step 5: Customize the Natural Web I/O Interface Server

In order to dispatch the NWO Natural sessions in CICS, you must adapt the configuration file of your Natural Web I/O Interface server running under z/OS in batch mode. For this purpose, two sample JCL members (`NWOIO09C` and `NWOCONFC`) are available.

Refer to *Configuring the Natural Web I/O Interface Server CICS Adapter* and to *Configuring the Natural Web I/O Interface Server*.

10 Configuring the Natural Web I/O Interface Server CICS Adapter

This chapter describes how to configure the CICS connection for a Natural Web I/O Interface server (product code NWO) running on z/OS.

Configuration File

After the installation of the Natural Web I/O Interface Server CICS Adapter is complete, the configuration of the Natural Web I/O Interface Server CICS Adapter has to be done in the configuration file of the corresponding Natural Web I/O Interface server.

To enable the CICS Adapter, specify the remote front-end module in the NWO configuration parameter `FRONTEND_NAME` (`FRONTEND_NAME=NATCSRFE`).

Configuration Parameters

The following CICS-relevant configuration parameters exist:

- RFE_CICS_TA_NAME
- RFE_CICS_FE_NAME
- RFE_CICS_TA_HOST
- RFE_CICS_TA_PORT
- RFE_CICS_TA_INIT_TOUT
- RFE_CICS_KEEP_TA
- RFE_CICS_TRACE

RFE_CICS_TA_NAME

This configuration parameter specifies the CICS transaction to be used for starting the remote front-end in CICS. This transaction must be defined in CICS and must refer to the program `NATCNRFE`. See also ***Installing the Natural Web I/O Interface Server CICS Adapter under z/OS***.

Note: At logon, this transaction name can be overridden by the user in order to switch to a different CICS transaction on a mainframe. See the section *Dynamically Changing the CICS Transaction Name when Starting a Session* in the client documentation for the Natural Web I/O Interface server.

Default Value	none
Example	`RFE_CICS_TA_NAME=NRFE`

RFE_CICS_FE_NAME

This configuration parameter specifies the Natural CICS nucleus you have installed with the applicable Natural for Mainframes installation under CICS. This program must be defined in CICS.

Default Value	none
Example	`RFE_CICS_FE_NAME=NCI`*vr*`NUC`

See also the Natural *Installation for Mainframes* documentation, *Installing the Natural CICS Interface*, *Customize CICS*.

RFE_CICS_TA_HOST

This configuration parameter specifies the TCP/IP address of the host the desired CICS is running. This parameter can be omitted if the Web I/O Interface server and CICS are running on the same TCP/IP node.

Default Value	The host address of the server.

Example	`RFE_CICS_TA_HOST=node1`
	or
	`RFE_CICS_TA_HOST=157.189.160.55`

RFE_CICS_TA_PORT

This configuration parameter specifies the TCP/IP port of the CICS supplied listener.

You can acquire this port number using the CICS supplied transaction EZAC. The CICS command `EZAC DISPLAY LISTENER` shows the definitions of the CICS standard listener.

Note: This port number is not used in the Web I/O Interface client to connect to a Web I/O Interface server. This port number (and the `RFE_CICS_TA_HOST` definition) is used internally by the Web I/O Interface server to communicate with the CICS region.

Possible Values	1 - 65535
Default Value	none
Example	RFE_CICS_TA_PORT=3010

RFE_CICS_TA_INIT_TOUT

If the Web I/O Interface client sends a request to a Natural Web I/O Interface server that is configured to use the CICS remote front-end, the remote front-end launches a CICS transaction (NRFE) for processing the request. The CICS transaction in turn listens to the TCP/IP to receive the data from the Web I/O Interface server required for processing the request.

This configuration parameter specifies the timeout value (in seconds) a launched transaction waits until the expected request data arrive from the server. If this timeout expires, the request aborts with a NAT9940 error.

Default Value	5
Example	RFE_CICS_TA_INIT_TOUT=20

Note: Do not define a value below 5.

RFE_CICS_KEEP_TA

For each request sent by Natural, the Natural Web I/O Interface server opens a TCP/IP connection to the CICS region and launches a CICS transaction (NRFE) for processing the request. With RFE_CICS_KEEP_TA=YES, the CICS transaction remains active for processing further requests of the same client. This saves the overhead for creating the TCP/IP connection and transaction initialization for successive requests, but consumes more resources within the CICS region due to waiting transactions.

The transaction wait time (for successive requests) is limited by RFE_CICS_TA_INIT_TOUT. That is, if the time slice between two successive requests exceeds the time specified by RFE_CICS_TA_INIT_TOUT, the CICS transaction and the TCP/IP connection is terminated independent of the RFE_CICS_KEEP_TA definition.

RFE_CICS_TA_INIT_TOUT=5 is a reasonable value to reuse transactions for multiple requests initiated by a single action in Natural Studio and to save CICS resources if Natural Studio waits for the next action of the user.

Default Value	NO
Example	RFE_CICS_KEEP_TA=YES

RFE_CICS_TRACE

This configuration parameter specifies the trace level for the remote front-end.

The trace level is similar to the trace implemented for the Web I/O Interface server. It is a bit string where each bit is responsible for a certain trace information:

Bit 31	Trace main events (transaction initialization/termination, request processing).
Bit 30	Detailed functions.
Bit 29	Dump internal storage areas.
Bit 27	Dump buffer header exchanged between Web I/O Interface server and CICS.
Bit 26	Dump entire buffer exchanged between Web I/O Interface server and CICS.
Bit 25	Dump the Natural Web I/O Interface server relevant buffer only (remote gateway buffer).
Bit 23	Trace error situations only.
Bit 07	Activate trace in the Web I/O Interface server region.
Bit 06	Activate trace in the CICS region.
Bit 00	Reserved for trace-level extension.

The trace destination is the data set defined for STDOUT.

Default Value	0
Example	RFE_CICS_TRACE=31+27+7 Dump main events and buffer header in the CICS region (Bits 31 + 27 + 7).

The following is a sample server configuration file using the Natural Web I/O Interface Server CICS Adapter:

```
   # The Web I/O Interface Server parameter.
SESSION_PARAMETER=PROFILE=(NWO,10,930)
FRONTEND_NAME=NATCSRFE         # Use the CICS Adapter front-end.
PORT_NUMBER=4711             # The port number used by the Web I/O Interface Client.

   # The CICS Adapter parameter.
RFE_CICS_TA_NAME=NRFE          # The CICS transaction for remote front-end.
RFE_CICS_TA_PORT=3010          # The port of the CICS listener.
                               # No RFE_CICS_TA_HOST is defined. This requires
                               # that CICS runs on the same node as the server.
RFE_CICS_FE_NAME=NCIvrNUC      # The name of the installed Natural CICS nucleus.
RFE_CICS_TA_INIT_TOUT=20       # Transaction timeout is 20 seconds.
```

Note: The server parameters `THREAD_NUMBER` and `THREAD_SIZE` are obsolete when the Natural Web I/O Interface Server CICS Adapter is used.

11 Installing the Natural Web I/O Interface Server IMS Adapter

This chapter describes how to install the IMS TM connection for a Natural Web I/O Interface server (abbreviated: “NWO server”) running under z/OS in batch mode.

Notation *vrs* or *vr*:

When used in this document, the notation *vrs* or *vr* represents the relevant product version (see also Version in the *Glossary*).

Prerequisites

For details, refer to the section *Prerequisites*.

Example Jobs

The example installation jobs are contained in the NWO*vrs*.JOBS data set and are prefixed with the product code. The data set is provided on the installation medium for Natural Web I/O Interface.

Installation Procedure

Perform the following steps:

Step 1: Customize the IMS Listener

See *z/OS Communications Server IP IMS Socket Guide* for the description of

- how to set up and operate the IMS Listener,
- how to use the IMS TCP/IP control statements.

Note: The port number of the IMS Listener is to be used later on in the NWO configuration parameter RFE_IMS_TA_PORT.

Step 2: Install the Server Transaction

It is assumed that the Natural IMS TM Interface is already installed. Thus, this step describes the additional steps which are necessary to create the Natural Web I/O Interface server environment.

Step 2.1: Adapt the Transaction Code Table

(Job I055, Steps 2555, 2556)

- In the `NTIMSPT` macro of the Natural parameter module, add the following entry:

```
        NTIMSPT TRAN=NATvrsAD,TYPE=SFE,                                 X
              ENVPID=ENVNW000,                                          X
              PSB=NIIvrsAD
```

- Assemble and link the parameter module.

Step 2.2: Adapt the Environment Parameters

(Job I055, Steps 2580, 2581)

- In the `NTIMSPE` macro of the Natural parameter module, add a new entry:

```
        NTIMSPE ENTRYNM=ENVNW000,                                       X
              THBELOW=OFF,                                              X
              THSIZE=1536000     (1500K)
```

- Assemble and link the parameter module.

Step 2.3: Relink the Natural IMS TM Interface Module

(Job I055, Step 2582)

Relink the Natural IMS TM interface module `NIIvrsIF`.

Step 2.4: Natural IMS TM Parameter Module

(Job I080, Steps 2500, 2510)

- Build the Natural IMS TM parameter module.
- Assemble and link the Natural IMS TM parameter module.

Step 2.5: Link the Server Front-End

(Job I080, Step 2586)

Link the server front-end.

Ensure that the name of the server front-end matches the value specified with the `PSB` keyword subparameter of the `NTIMSPT` macro in installation step 2.1 above.

Step 3: Customize IMS TM

- Define the PSB for the server transaction.

  As a minimum requirement you need a TP PCB.

  Define the application of the server transaction in the stage1 input. The transaction is a single mode, single segment non-conversational transaction.

  Example:

```
        APPLCTN PSB=NIIvrsAD,PGMTYPE=TP,SCHDTYP=PARALLEL
          TRANSACT CODE=NATvrsAD,MODE=SNGL,                          X
              MSGTYPE=(SNGLSEG,NONRESPONSE,4)
```

- Adapt the MPP region.

  In the JCL of the MPP region, add the `SYSTPCD` DD statement.

  See *z/OS Communications Server IP IMS Socket Guide* for the description on how to configure TCPIP.DATA.

- Concatenate the data set containing the NWO load library to the `STEPLIB` DD statement.

Before starting the Natural Web I/O Interface Server IMS Adapter installation, set the SMA parameter `NWO-SRV-IMS` to `Y` (yes).

To install the Natural Web I/O Interface Server IMS Adapter, perform the following steps:

Step 4: Create Member for NWO Server

(Job I009, Step 9412)

- Create member `NWOCONFI` for the NWO server.

Step 5: Link the Object Modules into the NWO Load Library

(Job I054, Step 9440)

Notes:

1. The module `NATINRFE` applies to two different products: the Natural Development Server (NDV) and the Natural Web I/O Interface (NWO server). So if you have already installed NDV, the module `NATINRFE` might already be there. However, it does not matter if you reinstall `NATINRFE` with NWO because the resulting module from either installation is the same.
2. The module `NATISRFE` (referenced in config via `FRONTEND_NAME=NATISRFE`) has already been linked in the prior steps of batch installation.

Step 6: Create Startup Procedure

(Job I200, Step 9417)

- Create the startup procedure for the NWO server.

Step 7: Customize the Natural Web I/O Interface Server

In order to dispatch the NWO Natural sessions in IMS, you must adapt the configuration file of your development server running under z/OS in batch mode. For this purpose, two sample JCL members (`NWOI009I` and `NWOCONFI`) are available.

Refer to *Configuring the Natural Development Server IMS Adapter* and to *Configuring the Natural Web I/O Interface Server*.

12 Configuring the Natural Web I/O Interface Server IMS Adapter

This chapter describes how to configure the IMS TM connection for a Natural Web I/O Interface server (product code NWO) running on z/OS.

Configuration File

After the installation of the Natural Web I/O Interface Server IMS Adapter is complete, the configuration of the Natural Web I/O Interface Server IMS Adapter has to be done in the configuration file of the corresponding Natural Web I/O Interface server.

To enable the Natural Web I/O Interface Server IMS Adapter, specify the remote front-end module in the NWO configuration parameter `FRONTEND_NAME`:

```
FRONTEND_NAME=NATISRFE.
```

Configuration Parameters

The following IMS-relevant configuration parameters exist:

- RFE_IMS_TA_NAME
- RFE_IMS_TA_HOST
- RFE_IMS_TA_PORT
- RFE_IMS_TRACE

RFE_IMS_TA_NAME

This configuration parameter specifies the IMS transaction to be used for starting the remote front-end in IMS TM. This transaction must be defined in IMS TM and must refer to the program `NATINRFE`. See also ***Installing the Natural Web I/O Interface Server IMS Adapter under z/OS***.

Default Value	none
Example	`RFE_IMS_TA_NAME=NAT`*`vrs`*`AD`

RFE_IMS_TA_HOST

This configuration parameter specifies the TCP/IP address of the host the desired IMS TM is running. This parameter can be omitted if the Web I/O Interface server and IMS TM are running on the same TCP/IP node.

Default Value	The host address of the server.

Example	`RFE_IMS_TA_HOST=node1`
	
	or
	
	`RFE_IMS_TA_HOST=157.189.160.55`

RFE_IMS_TA_PORT

This configuration parameter specifies the TCP/IP port of the IMS supplied listener.

Note: This port number is not used in the Web I/O Interface client to connect to a Web I/O Interface server. This port number (and the `RFE_IMS_TA_HOST` definition) is used internally by the Web I/O Interface server to communicate with the IMS region.

Possible Values	`1 - 65535`
Default Value	none
Example	`RFE_IMS_TA_PORT=3010`

RFE_IMS_TRACE

This configuration parameter specifies the trace level for the remote front-end.

The trace level is similar to the trace implemented for the Web I/O Interface server. It is a bit string where each bit is responsible for a certain trace information:

Bit 31	Trace main events (transaction initialization/termination, request processing).
Bit 30	Detailed functions.
Bit 29	Dump internal storage areas.
Bit 27	Dump buffer header exchanged between Web I/O Interface server and IMS TM.
Bit 26	Dump entire buffer exchanged between Web I/O Interface server and IMS TM.
Bit 25	Dump the Natural Web I/O Interface server relevant buffer only (remote gateway buffer).
Bit 23	Trace error situations only.
Bit 07	Activate trace in the Web I/O Interface server region.
Bit 06	Activate trace in the IMS MPP region.
Bit 00	Reserved for trace-level extension.

The trace destination is the data set defined for STDOUT.

Default Value	0
Example	RFE_IMS_TRACE=31+27+7 Dump main events and buffer header in the IMS MPP region (Bits 31 + 27 + 7).

The following is a sample server configuration file using the Natural Web I/O Interface Server IMS Adapter:

```
   # The Web I/O Interface Server parameter.
SESSION_PARAMETER=PROFILE=(NWO,10,930)
FRONTEND_NAME=NATISRFE         # Use the IMS Adapter front-end.
PORT_NUMBER=4711             # The port number used by the Web I/O Interface Client.

   # The IMS Adapter parameter.
RFE_IMS_TA_NAME=NATvrsAD       # The IMS transaction for the remote front-end.
RFE_IMS_TA_PORT=3020           # The port of the IMS listener.
                               # No RFE_IMS_TA_HOST is defined. This requires
                               # that IMS runs on the same node as the server.
```

Note: The server parameters THREAD_NUMBER and THREAD_SIZE are obsolete when the Natural Web I/O Interface Server IMS Adapter is used.

V Installing the Natural Web I/O Interface Client

This part explains how to install the Natural Web I/O Interface client on Tomcat so that it can be used with the server part of the Natural Web I/O Interface that is running in a Natural for Mainframes, Natural for Linux or Natural for Windows runtime environment.

The following topics are covered:

13 Prerequisites

Servlet Container

The following servlet container is supported. The servlet container is not delivered with the Natural Web I/O Interface. It can be obtained from the location indicated below, according to its license terms.

- Apache Tomcat 9 and 10 (see ***http://tomcat.apache.org/***).

Natural for Mainframes

If you want to use the Natural Web I/O Interface client with Natural for Mainframes, the following must be installed:

- Natural for Mainframes Version 8.2.5 or above, and
- the Natural Web I/O Interface server.

For detailed information, see:

- the *Installation* documentation for the different operating systems which is provided for Natural for Mainframes;
- the section ***Installing and Configuring the Natural Web I/O Interface Server*** in the version of this *Natural Web I/O Interface* documentation which is provided for Natural for Mainframes.

Natural for Linux

If you want to use the Natural Web I/O Interface client with Natural for Linux and Cloud, the following must be installed:

- Natural on Linux and Cloud Version 9.2.1 or above, and
- the Natural Web I/O Interface server and daemon.

For detailed information, see:

- the *Installation* documentation which is provided for Natural for Linux;
- the section *Installing and Configuring the Natural Web I/O Interface Server* in the version of this *Natural Web I/O Interface* documentation which is provided for Natural for Linux.

Natural for Windows

If you want to use the Natural Web I/O Interface client with Natural for Windows, the following must be installed:

- Natural for Windows Version 9.2.1 or above, and
- the Natural Web I/O Interface server and service.

For detailed information, see:

- the *Installation* documentation which is provided for Natural for Windows;
- the section *Installing and Configuring the Natural Web I/O Interface Server* in the version of this *Natural Web I/O Interface* documentation which is provided for Natural for Windows.

Browser Prerequisites

Supported browsers in this version are:

Internet Explorer 11 (deprecated)
Microsoft Edge (Chromium)
Mozilla Firefox Extended Support Release 91[(1)]
Google Chrome [(2)]

Notes:

[(1)] Only the Extended Support Releases of Mozilla Firefox are explicitly supported. Due to frequent upgrades of the Mozilla Firefox consumer release, the compatibility of Natural Web I/O Interface client with future versions of Mozilla Firefox cannot be fully guaranteed. Possible incompatibilities will be removed during the regular maintenance process of Natural Web I/O Interface client.

[(2)] The Google Chrome support is based on Google Chrome Version 115. Due to frequent version upgrades of Google Chrome, compatibility of Natural Web I/O Interface client with future versions of Google Chrome cannot be fully guaranteed. Possible incompatibilities will be removed during the regular maintenance process of Natural Web I/O Interface client.

Important: Cookies and JavaScript must be enabled in the browser.

14 Installing the Natural Web I/O Interface Client on Apache Tomcat

If you want to use the Natural Web I/O Interface client with Apache Tomcat, you must proceed as described in this chapter.

Installation Steps

The Natural Web I/O Interface client is installed using the Tomcat Manager.

The following is assumed:

- *<install-dir>* is the path to Software AG's main installation directory. By default, this is *C:\SoftwareAG* on Windows and */opt/softwareag* on Linux.
- *<host>* is the name of the machine on which Apache Tomcat is installed.
- *<port>* is the name of the port where Apache Tomcat is installed. In a default installation, this is port 8080.
- *<tomcat>* is the path to the directory in which Apache Tomcat is installed.

The following topics are covered below:

First-time Installation

To install the Natural Web I/O Interface client

1 Natural for Windows and Linux: Copy the complete contents of the *<install-dir>/natural/INSTALL/nwoclient/tomcat* directory to a directory of your choice on your hard disk.

2 Make sure that Apache Tomcat is running.

3 Open your web browser and enter the following URL:

```
http://<host>:<port>/manager/html
```

This opens the Tomcat Manager.

4 Deploy the web application file *natuniweb.war*:

- Under **Select WAR file to upload** select the path to the file *natuniweb.war*.
- Choose **Deploy**.

5 In the Tomcat Manager, look for the application **Natural Web I/O Interface Client** and choose **Reload**.

Update Installation

To update the Natural Web I/O Interface client

1 Natural for Windows and Linux: Copy the complete contents of the *<install-dir>/natural/INSTALL/nwoclient/tomcat* directory to a directory of your choice on your hard disk.

Or:

Download the Natural Web I/O Interface client for Apache Tomcat from Empower (***https://empower.softwareag.com/***) and unzip the contents to a directory of your choice on your hard disk.

2 Shut down Apache Tomcat.

3 Create a backup copy of your *sessions.xml* file, which is located in `<tomcat>`*/webapps/natuniweb/WEB-INF*.

4 Start Apache Tomcat.

5 Open your web browser and enter the following URL:

```
http://<host>:<port>/manager/html
```

This opens the Tomcat Manager.

6 Select *natuniweb.war* in the list of installed applications.

7 Choose **Undeploy**.

8 Deploy the new version of the Natural Web I/O Interface client as in a first-time installation.

9 Restore the *sessions.xml* file that you have backed up previously.

Installation Verification

It is assumed that *http://<host>:<port>* is the URL of your application server.

To verify the installation

- Enter the following URL in your web browser:

```
http://<host>:<port>/natuniweb/natural.jsp
```

For example:

```
http://myhost:8080/natuniweb/natural.jsp
```

The Natural Web I/O Interface client is now started in your browser. The entries which appear in the resulting logon page depend on the settings in your configuration file. For further information, see *Configuring the Client*.

15 Migrating the Natural Web I/O Interface Client from IIS to Apache Tomcat

Microsoft Internet Information Services (IIS) is no longer supported. If you are currently using the Natural Web I/O Interface client on IIS, you have to move to Apache Tomcat.

Note: JBoss Application Server and Oracle GlassFish Server are also no longer supported. If you are currently using one of these application servers, you also have to move to Apache Tomcat. In this case, however, you can reuse your previous settings (that is, the URL for logon page and the configuration file *sessions.xml*).

The most simple solution is to migrate the Natural Web I/O Interface client from IIS to Apache Tomcat. Therefore, this chapter gives IIS administrators a quick introduction to a Tomcat installation and describes the migration steps.

Before You Install the Natural Web I/O Interface Client

If Apache Tomcat is not yet installed, proceed as described in the topics below:

- Installing Tomcat
- Installing Java
- Starting the Tomcat Server

Installing Tomcat

Go to ***http://tomcat.apache.org/*** and download Tomcat as a zip file.

For Microsoft Windows users: download either the 32-bit or the 64-bit Windows zip file.

Unzip the downloaded zip file to a directory of your choice.

Installing Java

Tomcat is based on Java. Therefore, you have to make sure that a Java Runtime Environment (JRE) or a Java Development Kit (JDK) is installed. The version of the Java runtime should be at least Java 6 update 24. This is the minimum version that is required for the Natural Web I/O Interface client on Tomcat.

You can download the Java JRE or JDK from the Oracle website at ***http://www.oracle.com/technetwork/java/javase/downloads/index.html***.

If Java is installed on your system, make sure that the environment variable `JAVA_HOME` is set to the Java home directory.

Starting the Tomcat Server

When Tomcat and the appropriate Java version have been installed, you can start Tomcat.

To start Tomcat, execute the *startup.bat* file from the *bin* directory of your Tomcat installation. To check whether Tomcat is running, enter the following URL:

```
http://localhost:8080
```

This should display Tomcat's default home page.

Installing the Natural Web I/O Interface Client on Apache Tomcat

When Apache Tomcat has been installed, install the Natural Web I/O Interface client as described in ***Installing the Natural Web I/O Interface Client on Apache Tomcat***.

Configuring the Natural Web I/O Interface Client on Apache Tomcat

When the Natural Web I/O Interface client has been installed, proceed as described in the topics below:

- Invoking the Logon Page
- Changing the Tomcat HTTP Port
- Using the Settings from Your IIS Configuration File
- Using the Configuration Tool
- Protecting the Configuration Tool Against Unauthorized Access
- Displaying the Logon Page by Default

Invoking the Logon Page

Enter the following URL to invoke the logon page (this is different from the URL that was used with IIS):

```
http://localhost:8080/natuniweb/natural.jsp
```

Changing the Tomcat HTTP Port

IIS usually runs on the default port 80. If you want Tomcat to work with the same port, edit the file *server.xml* which is located in Tomcat's *conf* subdirectory and then search for the following text:

```
<Connector port="8080" protocol="HTTP/1.1"
```

Change the port number so that it looks as follows:

```
<Connector port="80" protocol="HTTP/1.1"
```

Using the Settings from Your IIS Configuration File

With Tomcat, you can reuse the *settings.xml* configuration file of IIS, but you have to rename the file to *sessions.xml*. Proceed as follows:

1. Copy the *settings.xml* file from your IIS installation to the following directory of your Tomcat installation:

   webapps/natuniweb/WEB-INF
2. Either rename the *sessions.xml* file which comes with the Natural Web I/O Interface client installation on Tomcat (for example, to *sessions-original.xml*) or delete it.
3. Rename the *settings.xml* file to *sessions.xml*.

Using the Configuration Tool

When the Natural Web I/O Interface client runs on Tomcat, it is no longer necessary to edit the configuration file manually. Instead, you can use the configuration tool. Using this tool has the advantage that it is not possible for you to create invalid XML code and thus damage the XML file. See ***Using the Configuration Tool*** for further information.

The IIS-specific entries in the renamed configuration file will be ignored. These are:

```
natural_parameter visible
theme
screen top
screen left
screen size
screen pfkeypos
```

You can still edit the configuration file manually. However, this is no longer recommended.

Protecting the Configuration Tool Against Unauthorized Access

It is possible to protect the configuration tool against unauthorized access. See *Configuring Container-Managed Security* for detailed information.

For detailed information on the necessary realm configuration for Tomcat, see *http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html*.

Displaying the Logon Page by Default

When you enter the URL *http://localhost:8080/natuniweb*, Tomcat shows the default page of the Natural Web I/O Interface client which allows you to access either the **logon page** or the **configuration tool** of the Natural Web I/O Interface client.

Note: If you have defined a different port (for example, 80), make sure to use that port number in the URL.

This behavior is different from IIS which displays the logon page by default. If you also want Tomcat to display the logon page by default, edit the file *web.xml* which is located in Tomcat's *webapps\natuniweb\WEB-INF* directory and search for the following entry:

```
<welcome-file-list>
  <welcome-file>
         index.html
  </welcome-file>
</welcome-file-list>
```

Change the name of the welcome file to *natural.jsp* as shown in the following example:

```
<welcome-file-list>
  <welcome-file>
         natural.jsp
  </welcome-file>
</welcome-file-list>
```

VI Configuring the Client

This part explains how to configure the Natural Web I/O Interface client so that it can be used in a Natural runtime environment. The following topics are covered:

16 About the Logon Page

Starting a Natural Application from the Logon Page

When you start the Natural Web I/O Interface client in the browser, a logon page appears. The entries in this logon page depend on the settings in your **Session Configuration**.

In order to start a Natural application from the logon page, you enter the following URL inside your browser:

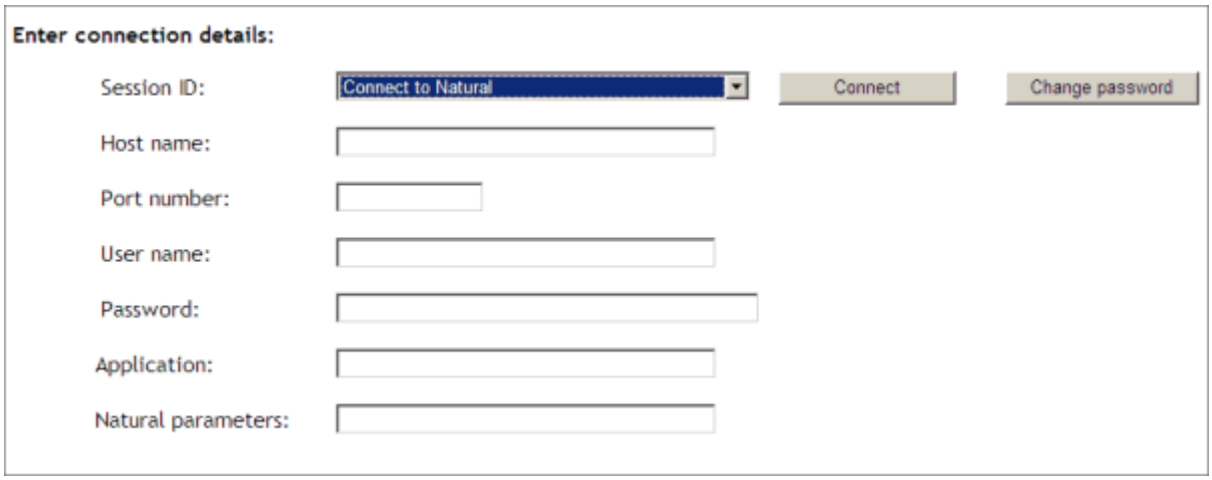
```
http://<host>:<port>/natuniweb/natural.jsp
```

where *<host>* and *<port>* are the host name and port number of your application server.

Examples of Logon Pages

For each session definition that has been configured in the configuration file, an entry appears on the logon page. If the user selects the corresponding entry, only those parameters that were not pre-configured in the configuration file need to be specified in the logon page in order to start the application. Usually, you will preconfigure all connection parameters except user name and password.

The following example shows part of a logon page which results from a configuration file in which no special entries are defined for a session:



The following example shows part of a logon page which results from a configuration file in which many settings are already predefined (including user ID and password):

To log on to a session, you have to specify all required information in the logon page (for example, you select a session from the corresponding drop-down list box). When you choose the **Connect** button, the screen for the selected session appears.

Dynamically Changing the CICS Transaction Name when Starting a Session

The following description applies if you want to switch to a different CICS transaction on a mainframe.

You specify the CICS transaction name in the same text box in which you also specify the dynamic parameters for the Natural environment. So that the CICS transaction name can be evaluated, it is important that you specify it before any Natural parameters, using the following syntax:

```
<TA_NAME=name>
```

where *name* can be 1 to 4 characters long. This must be the name of an existing CICS transaction which applies to a CICS Adapter. It will override the transaction name which is currently defined in the configuration file for the CICS Adapter on the Natural Web I/O Interface server (NWO) . Ask your administrator for further information.

Make sure to put the entire definition in angle brackets. When this definition is followed by a Natural parameter, insert a blank before the Natural parameter. Example:

```
<TA_NAME=NA82> STACK=(LOGON SYSCP)
```

If the specified CICS transaction name cannot be found, an error message occurs and the session cannot be started.

Note: The above definition for the CICS transaction name can also be specified in the **configuration tool** , in the same place where you also specify the Natural parameters, and together with the **URL parameter** `natparam` .

Specifying a Password in the Logon Page

The following information applies when the field for entering a password appears on the logon page. This field does not appear when a password has already been defined in the configuration file.

Under Windows and Linux, you always have to enter the operating system password, even if Natural Security is active.

On the mainframe, this is different: When Natural Security is not active, you have to enter the operating system password. When Natural Security is active, you have to enter the Natural Security password.

Changing the Password in the Logon Page

Currently, this functionality is only available for Natural for Linux and Natural for Windows.

The following information applies when the fields for entering a user ID and a password appear on the logon page. These fields do not appear when user ID and password have already been defined in the configuration file; in this case, it is not possible to change the password in the logon page.

When your password has expired, you are automatically asked for a new password. When you try to log on with your current password, an error message appears and input fields for changing the password are shown.

To change the password

1 Choose the **Change password** button in the logon page.

The name of this button changes to **DonÃ¢Â€Â™t change password** and the following two input fields are shown in the logon page:

- **New password**
- **Repeat new password**

2 Enter your user ID and your current password as usual.

3 Enter the new password in the two input fields.

4 Choose the **Connect** button to change the password.

Or:

If you do not want to change your password, choose the **DonÃ¢Â€Â™t change password** button. The two input fields will then disappear.

Browser Restrictions

The browser's “Back” and “Forward” buttons do not work with the Natural Web I/O Interface client and should therefore not be used.

If you want to run two Natural sessions in parallel, you have to start a new instance of the browser (for example, by choosing the corresponding icon in the Quick Launch toolbar of Windows). You must not use the browser's “New Window” function. This would result in one session running in two browsers, which is not allowed.

17 Natural Client Configuration Tool

Invoking the Configuration Tool

The Natural Web I/O Interface client offers a configuration tool. The configuration tool is used to create the session configurations which are then available in the logon page. It can also be used for logging purposes in case of problems; however, this should only be done when requested by Software AG support.

The configuration tool is automatically installed when you install the Natural Web I/O Interface client .

To invoke the configuration tool

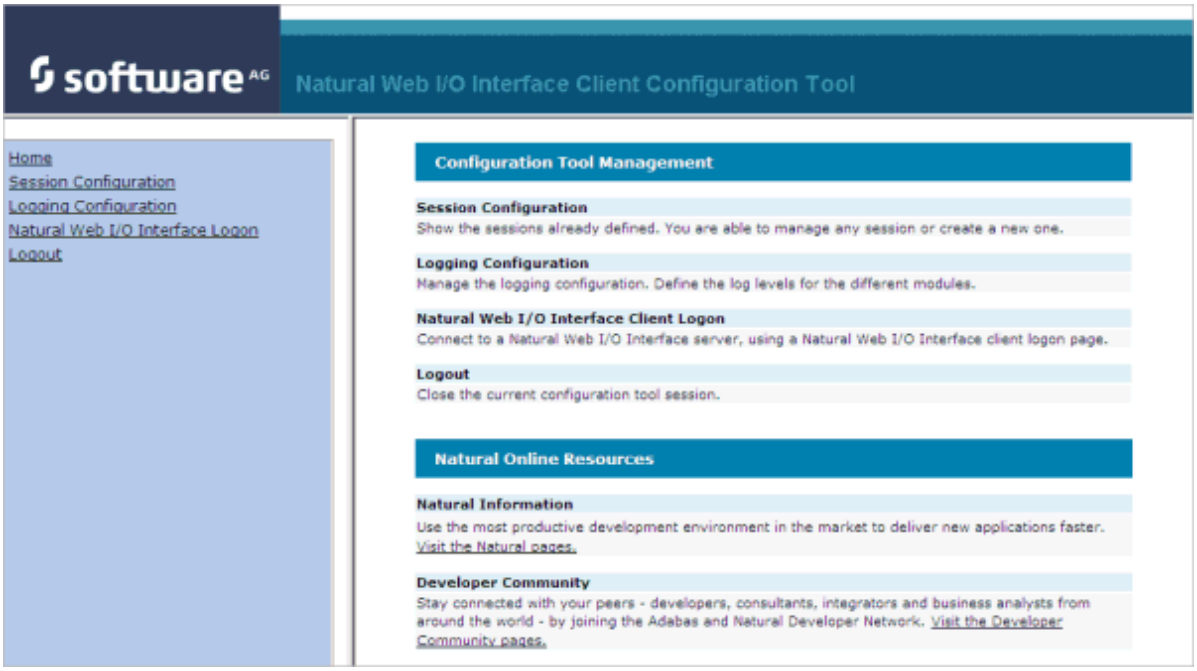
- Enter the following URL in your browser:

  ```
  http://<host>:<port>/natuniweb/conf_index.jsp
  ```

  where *<host>* and *<port>* are the host name and port number of your application server.

  Note: You might wish to protect the configuration tool against unauthorized access. See *Configuring Container-Managed Security* for information on how to restrict the access to sensitive areas of the application server environment. If you have restricted access to the configuration tool, an authentication dialog appears. The appearance of this dialog depends on the authentication model you have chosen.

  The configuration tool appears.



The configuration tool has two frames.

The home page of the configuration tool is initially shown in the right frame. It provides brief descriptions for the links provided in the left frame. It also provides links to several Software AG pages on the web.

When you have invoked a function (for example, when you are currently viewing the session configuration), you can always choose the **Home** link in the left frame to return to the home page of the configuration tool.

The functions that are invoked by the other links in the left frame are described below.

Session Configuration

This section explains how to manage the content of the configuration file for the sessions. It covers the following topics:

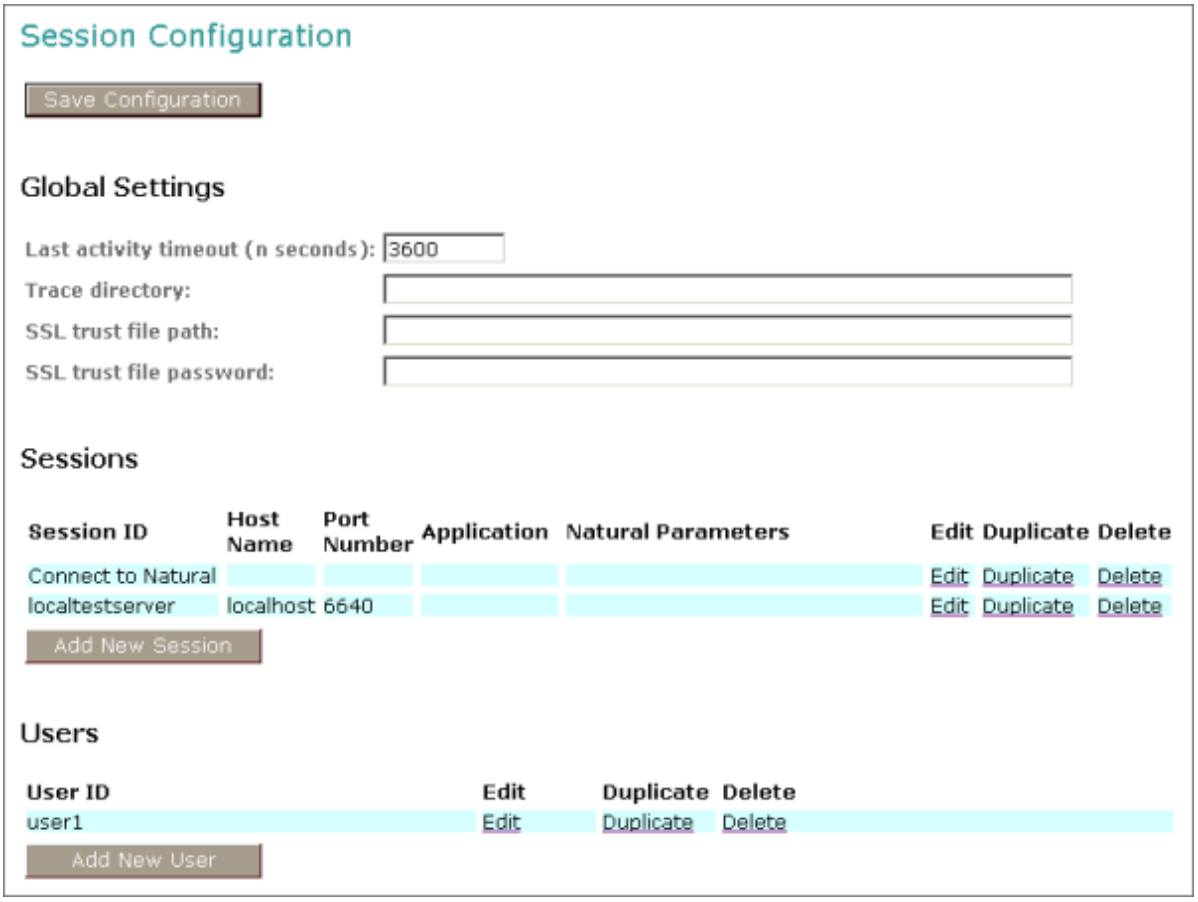
Invoking the Session Configuration Page

The content of the configuration file for the sessions is managed using the **Session Configuration** page.

To invoke the Session Configuration page

- In the frame on the left, choose the **Session Configuration** link.

  The **Session Configuration** page appears in the right frame. It shows the global settings and lists all sessions and users that are currently defined. For a session, some of the configuration file information is shown. Example:



Global Settings

The global settings apply for all defined sessions. You can define the following global settings in the configuration file:

Option	Description
Last activity timeout (n seconds)	The number of seconds that the client waits for the next user activity. When the defined number of seconds has been reached without user activity, the session is closed. The default is 3600 seconds.
Trace directory	Optional. Location of a different trace directory. When a different trace directory is not defined, the trace files are written to the default trace directory. By default, the trace files are written to the directory which has been set by the Java property `java.io.tmpdir`. On Windows, this is normally the environment variable `TMP` for the user who started the application server. On Linux, this is normally */tmp* or */var/tmp* . You can also set this property in the start script for the application server.

Option	Description
	Tracing can be enabled individually for each session (see *Overview of Session Options* below). However, it should only be enabled when requested by Software AG support.
SSL trust file path	Optional. The path to your trust file. See *Configuring SSL* for further information.
SSL trust file password	If your trust file is password-protected, you have to specify the appropriate password. When you do not specify the password for a password-protected trust file, the trust file cannot be opened and it is thus not possible to open an SSL session. When your trust file is not password-protected, you should not specify a password.

Adding a New Session

You can add a new session to the configuration file.

To add a new session

1 Choose the **Add New Session** button.

The **Edit Session** page appears.

2 Specify all required information as described below in the section *Overview of Session Options* .

3 Choose the **OK** button to return to the **Session Configuration** page.

The new session is not yet available in the configuration file.

4 Choose the **Save Configuration** button to write the new session to the configuration file.

Editing a Session

You can edit any existing session in the configuration file.

To edit a session

1 Choose the **Edit** link that is shown next to the session that you want to edit.

The **Edit Session** page appears.

2 Specify all required information as described below in the section *Overview of Session Options* .

3 Choose the **OK** button to return to the **Session Configuration** page.

The modifications are not yet available in the configuration file.

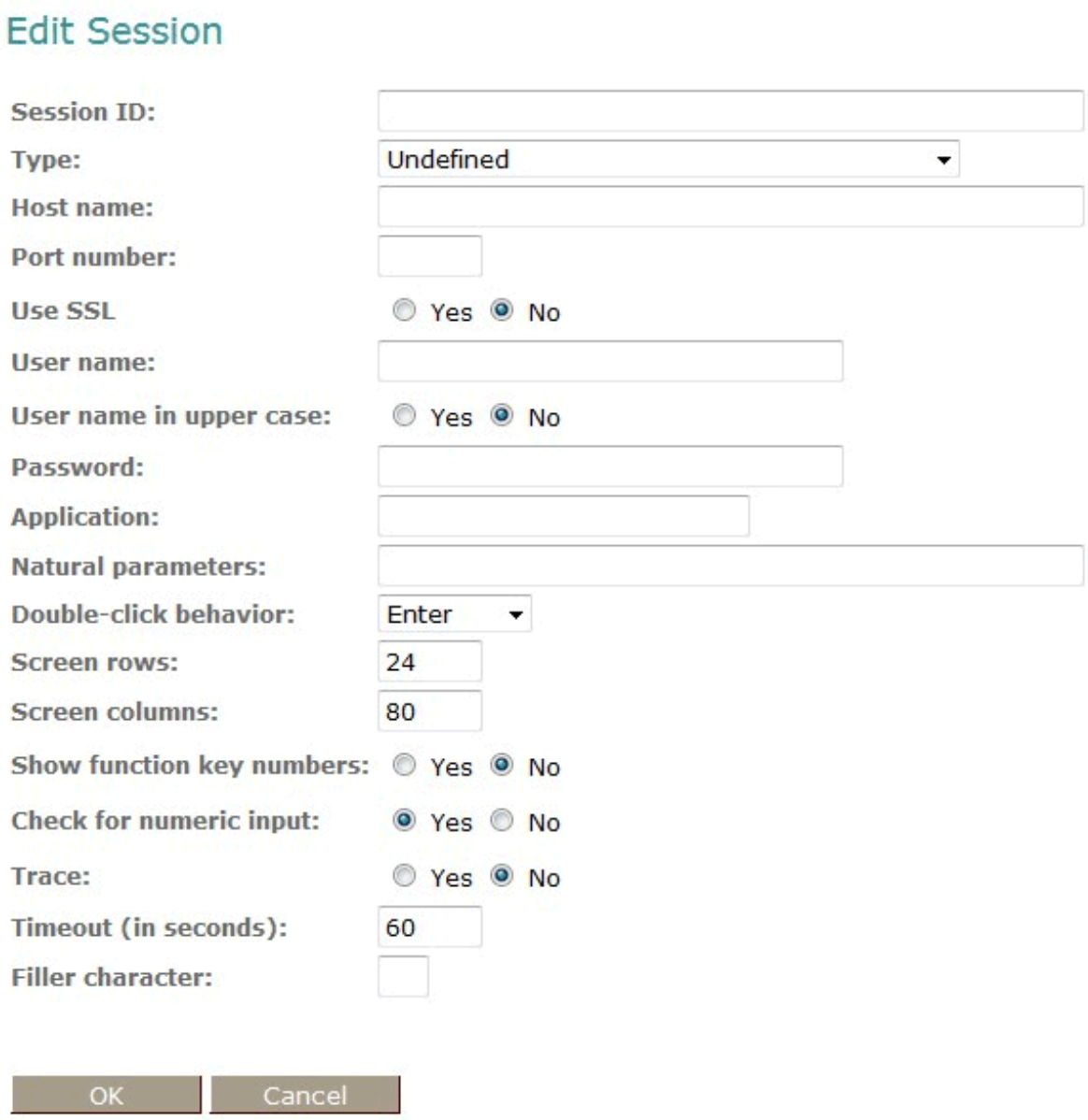
4 Choose the **Save Configuration** button to write the modifications to the configuration file.

Overview of Session Options

The **Edit Session** page appears when you

- **add** a new session, or
- **edit** an existing session.

Example:



The **Edit Session** page provides the following options:

Option	Description
Session ID	Mandatory. A session name of your choice. On the logon page, the session name is provided in a drop-down list box.
Type	The platform on which user ID and password are authenticated. You can select the required setting from the drop-down list box. ■ **Undefined** Default. User ID and password can have a maximum of 32 characters. See also the description for Natural for Windows or Linux below. ■ **Natural for Mainframes** User ID and password can have a maximum of 8 characters. ■ **Natural for Mainframes with Natural Security** User ID and password can have a maximum of 8 characters. The user ID must comply with the Natural naming conventions for library names . ■ **Natural for Windows or Linux** User ID and password can have a maximum of 32 characters. When a domain is required, you have to specify it together with the user ID (in the form "*domain**user-ID*").
Host name	The name or TCP/IP address of the server on which Natural and the Natural Web I/O Interface server are running. When this is specified, the corresponding field does not appear on the logon page.
Port number	The TCP/IP port number on which the Natural Web I/O Interface server is listening. When this is specified, the corresponding field does not appear on the logon page.
Use SSL	If set to **Yes** , a secure connection is established between the Natural Web I/O Interface client on the application server and the Natural Web I/O Interface server. **Important:** If you want to use SSL with Natural for Mainframes, one of the corresponding mainframe types must be selected; the type must not be **Undefined** or **Natural for Windows or Linux** . The other way around, if you want to use SSL with Natural for Windows or Linux, you must not select one of the mainframe types; the type may also be **Undefined** in this case.
User name	Optional. A valid user ID for the current machine. When this is specified, the corresponding field does not appear on the logon page.
User name in upper case	If selected, the input field for the user ID is in upper-case mode.
Password	Optional. A valid password for the above user ID. Under Windows and Linux, this is always the operating system password of the user, even if Natural Security is active. On the mainframe, this is different: When Natural Security is not active, this is the operating system password of the user. When Natural Security is active, this is the Natural Security password.

Option	Description
	When a password is specified, the corresponding field does not appear on the logon page. The configuration tool saves the password in encrypted form.
Application	■ **Natural for Mainframes** The name of the Natural program or a command sequence that starts your application as you would enter it on the NEXT prompt. Example: `TEST01 data1,data2` ■ **Natural for Linux** The name of the Linux shell script for starting the Natural application (a file similar to *nwo.sh*). ■ **Natural for Windows** The name of the Windows command file (*.bat*) for starting the Natural application. When this is specified, the corresponding field does not appear on the logon page.

Natural parameters	Optional. Parameters for starting the Natural application. This can be stack parameters, a parameter file/module or other Natural-specific information.
	■ **Natural for Mainframes**
	Used to pass dynamic Natural profile parameters to the session, for example:
	`SYSPARM=(MYPARMS) STACK=(LOGON MYAPPL)`
	Note: It is recommended to specify the Natural program that starts the application with the option **Application** instead of passing it with the profile parameter STACK .
	■ **Natural for Linux and Natural for Windows**
	Used when the above shell script (Linux) or command file (Windows) uses the parameter $5 after "natural" , for example:
	`PARM=MYPARM STACK=(LOGON MYLIB;MENU)`

Double-click behavior	The key that is to be simulated when double-clicking an output field. By default, this is the ENTER key. It is possible to disable the double-click behavior, or to define a function key (PF1 through PF12). You can select the required setting from the drop-down list box. **Tip:** When context-sensitive help has been defined for the output fields, it may be useful to define PF1 . The help function will then be invoked when the user double-clicks an output field.
Screen rows	The number of rows in the output window. Possible values: minimum 24, no upper limit. Default: 24. Not used by Natural for Mainframes which uses the profile parameter TMODEL instead.

Option	Description
Screen columns	The number of columns in the output window. Possible values: minimum 80, no upper limit. Default: 80. Not used by Natural for Mainframes which uses the profile parameter `TMODEL` instead.
Show function key numbers	If set to **Yes** , the PF key numbers are shown next to the PF keys.
Trace	Should only be set to **Yes** when requested by Software AG support.
Check for numeric input	If set to **Yes** (default), numeric input fields are validated. In this case, only the following characters are allowed in numeric input fields (in addition to the numbers "0" through "9"): *blank* + (plus) - (minus) _ (underscore) , (comma . (period) ? (question mark) If set to **No** , numeric input fields are not validated.
Timeout (in seconds)	The number of seconds that the client waits for a response after an updated page was sent to the Natural session. When the defined number of seconds has been reached without response, the session is closed. The default is 60 seconds. Normally, you need not change this value.
Filler character	Optional. The filler character that is to be removed from the input fields. An application can define, for example, an underscore (_) as the filler character. Trailing filler characters will be removed from the input fields, and leading filler characters will be replaced with blanks.

Duplicating a Session

You can add a copy of any existing session to the configuration file.

To duplicate a session

1 Choose the **Duplicate** link that is shown next to the session that you want to duplicate.

A new entry is shown at the bottom of the list of sessions. Its name is "Copy of *session-ID*" . The duplicated session is not yet available in the configuration file.

2 **Edit** and save the duplicated session as described above.

Deleting a Session

You can delete any existing session from the configuration file.

To delete a session

1 Choose the **Delete** link that is shown next to the session that you want to delete.

The session is deleted from the list of sessions. It is not yet deleted in the configuration file.

2 Choose the **Save Configuration** button to delete the session from the configuration file.

Adding a New User

You can predefine Natural users and their passwords in the configuration file.

When a Natural page is opened with a URL that specifies a user in the URL parameter `natuser`, the specified user is matched against the list of users in the configuration file. When the specified user is defined in the configuration file, the corresponding password is used to authenticate the user when the Natural session is started. See also *Starting a Natural Application with a URL*.

Example - when the following URL is used, the password defined for "user1" is used:

http://myhost:8080/natuniweb/natural.jsp?natuser=user1...

To add a new user

1 Choose the **Add New User** button.

The **Edit User** page appears.

2 Specify a user name and passwort

3 Choose the **OK** button to return to the **Session Configuration** page.

The new user is not yet available in the configuration file.

4 Choose the **Save Configuration** button to write the new user to the configuration file.

Note: You edit, duplicate and delete a user in the same way as a session (see the corresponding descriptions above).

Saving the Configuration

When you choose the **Save Configuration** button, all of your changes are written to the configuration file. The server picks up the new settings automatically the next time it reads data from the configuration file.

Caution: If you do not choose the **Save Configuration** button but log out instead or leave the configuration tool by entering another URL, the new settings are not written to the configuration file.

Logging Configuration

The content of the configuration file for logging is managed using the **Logging Configuration** page. See the section *Logging* for detailed information.

Logon Page

The configuration tool provides the following link in the left frame:

- **Natural Web I/O Interface Logon**

This link opens the logon page in the right frame.

The logon page uses the current settings in the configuration file. When you select a session from the drop-down list box, you can check whether the connection details are shown as desired. If not, you can go back to the session configuration and modify the settings of the corresponding session.

See also *About the Logon Page* .

Logout

When the configuration tool is protected against unauthorized access and you log out of the configuration tool, you make sure that no other user can change the client configuration when you leave your PC unattended for a while.

To log out

- In the frame on the left, choose the **Logout** link.

When the configuration tool is protected against unauthorized access, the authentication dialog is shown again.

When it is not protected, the home page is shown again.

18 Ajax Configuration

The following topics are covered below:

General cisconfig.xml Parameters

The *cisconfig.xml* file contains some general control information. The following is a very basic example:

```
<cisconfig startmonitoringthread="true"
           requestclienthost="false"
           debugmode="false"
           loglevel="EWI"
           logtoscreen="false"
           sessiontimeout="3600"
           xmldatamanager="com.softwareag.cis.xmldata.filebased.XMLDataManager"
           useownclassloader="true"
           browserpopuponerror="false"
           framebuffersize="3"
           ↩
onlinehelpmanager="com.softwareag.cis.onlinehelp.projectbased.FrameHelpOHManager"
           textencoding="UTF-8"
           enableadapterpreload="true">
</cisconfig>
```

`accessibilityroles`	Default: true Defines for controls and container that corresponding role attributes for accessibility are generated.
`animatecontrols`	Default: true. Defines how Application Designer handles the animation of controls. There are several controls that can be rendered in an animated way and in a standard way. Setting this parameter to "false" can help to improve performance, especially if you are not using the newest hardware. Values: true/false.
`buttonctrlenter`	Default false. If set to true <ctrl><enter> on a focused button will trigger the button method.
`browserpopuponerror`	Default: false. Defines how Application Designer handles it if the application behind an Application Designer page throws an error.

	By default (false), the browser switches to an error screen. In the screen, the user can only abort the current function. This is the default way in which any kind of inconsistency is automatically omitted. When you set `browserpopuponerror` to "true", the browser opens a pop-up window in which the error is output. This setting should only be used during development because it may cause inconsistencies in the application. Values: true/false.
`clientsideerrorinstatusbar`	Default: false. By default, client-side error messages are displayed as pop-ups. When you set this parameter to "true", client-side error messages are displayed in the status bar. Values: true/false.
`collectionorblocklimit`	Default: 300. Defines the maximum number of items in a grid after which the framework automatically switches from client-side scrolling to server-side scrolling.
`completedateinput`	Default: true. By default, partial input in the `DATEINPUT` control is automatically completed. When you set this parameter to "false", no automatic completion will be done, thus forcing end-users to always enter the complete date. Values: true/false.
`createhttpsession`	Default: false. Internally, Application Designer does not require HTTP session management that is provided by the servlet container. Some application servers (especially in clustered scenarios in which Application Designer runs in several nodes) require an explicit HTTP session ID to be used in order to route requests from a browser client always to the right application server node in the cluster. Set `createhttpsession` to "true" in this case. Values: true/false.
`debugmode`	Default: false. A log is written permanently into Application Designer's *log* directory. When `debugmode` is set to "true", a lot of information which normally is not required is written to the log.

	Be aware that you can also set the debug mode dynamically within your running system. Application Designer provides a monitoring tool in which you can switch the debug mode on and off. Values: true/false.
`defaultcss`	You can set your own default style sheet for your entire application. For example: `../cis/styles/MY_STYLE.css`
`defaultlanguage`	Default: en (English). Defines the language that is to be used by default when starting Application Designer. If not set, "en" is used.
`designtimeclassloader`	By default, Application Designer uses an own class loader for accessing adapter classes at design time. (You can switch this off by specifying `useownclassloader="false"`.) With the `designtimeclassloader`, you can explicitly select a class loader class that Application Designer is to use. This allows you to use class loaders that offer special functions such as reading encrypted class files. Value: the name of a class loader class.
`displayallowtab`	Default: true Defines if tabbing into DISPLAY input controls is possible.
`enableadapterpreload`	Default: true. By default, the server sends all required responses at once to the client, even if different adapters are involved. If set to "false", a separate data transfer occurs for each involved adapter.
`errorreactionadapter`	In case of an unhandled application error, the Application Designer runtime navigates to an error page. The class name specified in `errorreactionadapter` is the Java adapter for this error page. If an error reaction adapter is not specified, a default adapter is used which shows the error's stack trace. The Application Designer framework contains a second error reaction adapter with the class name `com.softwareag.cis.server.SecureErrorReactionAdapter`. For security reasons, this adapter does not show a stack trace but only an error message. You can write your own error reaction adapter and create your own error page. An error reaction adapter must implement one of the

	interfaces `com.softwareag.cis.server.ISecureErrorReactionAdapter` or `com.softwareag.cis.server.IErrorReactionAdapter`. For more information, see the corresponding Java documentation.
`fieldnumerictypesrightaligned`	Default: false. Set this parameter to "true" in order to right-align text within the FIELD control when using the data type `int`, `long` or `float`. Values: true/false.
`flushreceivespreviousfocused`	Default: false. By default, during a flush event the adapter gets as focus information the input control that *received* the focus. Set this parameter to "true" if during a flush event your application relies on getting as focus information the input control that *lost* the focus. For Natural applications this means: By default, the Natural system variable `*CURS-FIELD` contains during the flush event the value of the Natural system function `POS` for the input control that received the focus. Values: true/false.
`framebuffersize`	Default: 3. Each page in the browser client runs inside a surrounding page. This surrounding page offers a couple of internal functions, one of them to buffer contained Application Designer pages: if a user opens the first page and then navigates to a second page, the first page is internally kept inside a frame buffer. If returning to the first page later on, the browser does not have to build up the first page from scratch but just switches to the buffered page. The `framebuffersize` defines the number of buffered pages. Increasing the `framebuffersize` means that more resources are used on the client (browser) side. When changing this value, you should test the memory consumption on the client side before rolling out the change to productively running implementations. Value: integer number.
`htmlgeneratorlog`	Defaut: false. By default *.protocol* files are written during the HTML generation. If set to "true", an additional *.log* file is written for each layout. Only set this to "true", if you cannot resolve generation errors via the Layout Painter error marking. It reduces generation performance.
`Itrinlinedisplay`	Only set this if you notice rounding issues with pixel-sizing in ITRs while zooming the page in Google Chrome and/or Edge Chromium.

	For more information, see *ITR in Google Chrome and Edge Chromium in Natural* and Java.
`licwarningsfor`	Semicolon seperated list of hostnames. When the web application is called with an URL containing one of these hostnames and the license is in the expiration period of 40 days, an alert box with a license warning is shown once per day.
`loglevel`	Default: EWI. Defines the message types that are to be logged. Values: E (error) W (warning) I (information) D (debug) N (no logging) You can specify any combination of message types by concatenating the message types. Example: "EW" logs all error and warning messages. "EWI" additionally logs information messages. Specify "N" (no logging) to switch off writing log messages to a logfile. **Caution:** When having set `debugmode` to "true", the `loglevel` filter is automatically bypassed and all messages are logged. `debugmode` is stronger than `loglevel`.
`logtoscreen`	Default: false. If this parameter is set to "true", all Application Designer log information is also output to the command screen from which you started Application Designer. This parameter should only be set to "true" if running in development mode. Values: true/false.
`maxitemsinfieldcombo`	Default: 100. The FIELD control provides for a predefined pop-up method `openIdValueComboOrPopup`. Depending on the size of the list of valid values, the list is either shown in a combo box or in a pop-up. Use this parameter to control the maximum number of entries that are to be shown in the combo box. Value: integer number.
`maxserverlogage`	Default: -1 (log files are not automatically deleted). When setting `maxserverlogage` to a value > 0, *ServerLog*.log* files, which are older than the set number of days, are automatically deleted.

	For example, if `maxserverlogage` is set to 3, all *ServerLog*.log* files, which are older than 3 days are automatically deleted. If `startmonitoringthread` is set to false, this parameter has no effect.
`maxworkplaceactivities`	Default: -1 (unlimited). The maximum number of workplace activities in a workplace application.
`monitoringthreadinterval`	Default: 5000. The interval in milliseconds for the wake-up of the monitoring thread. If `startmonitoringthread` is set to false, this parameter has no effect.
`multilanguagemanager`	Internally, Application Designer uses an interface to retrieve the translation information for a certain text ID and a certain language. A default implementation is available that stores the corresponding language information in files that are part of the web application. Value: the name of the class that supports Application Designer's multi-language interface.
`natuppercase`	Default: false. Set this parameter to "true" if your Natural program only allows Latin upper-case characters. This is the case, for example, if your Natural program uses the Hebrew codepage CP803. **Important:** Set the parameter `natuppercase="true"` *before* you implement your main program with Natural for Ajax. If you set this parameter after the implemention, you will have to change all Latin lower-case characters to upper-case manually. Values: true/false.
`notifyparentonpopupclosed`	Default: true. If this parameter is set to "false", no `nat:page.default` will be sent to the parent Natural program after a pop-up is closed. Otherwise the parent Natural program will receive a `nat:page.default` event after a pop-up is closed. Values: true/false.
`onlinehelpmanager`	Application Designer accesses a certain URL when the user presses F1 on certain controls (for example, fields, check boxes and others). Application Designer transfers a corresponding help ID that is defined with the control into a URL and opens this URL in a pop-up window. If you have your own mechanisms for defining this URL, you can implement a corresponding Application Designer Java interface (`com.softwareag.cis.onlinehelp.IOHManager`).

	Value: the name of the interface.
pagepopupenterhotkey	Default: false. By default, the reactOnPagePopupEnterKey event is not triggered when ENTER is pressed in the page pop-up. When setting this parameter to "true", the event reactOnPagePopupEnterKey is triggered when ENTER is pressed in the page pop-up. This event can be processed in the Natural program. Values: true/false.
pagepopuphorizontal	Use this to automatically adapt the size of a page pop-up if it does not fit to its parent because the parent width is not big enough. Supported values are "zoom" and "resize". If set to "resize" the width of a page pop-up is reduced. If set to "zoom" the page pop-up will automatically be zoomed in. Values: resize/zoom.
pagepopuponresize	Default: false. If set to true an open page pop-up is resized when it's parent is resized. Values: true/false.
pagepopupvertical	Use this to automatically adapt the size of a page pop-up if it does not fit to its parent because the parent height is not big enough. Supported values are "zoom" and "resize". If set to "resize" the height of a page pop-up is reduced. If set to "zoom" the page pop-up will automatically be zoomed in. Values: resize/zoom.
popupparentdisabled	Default: false. When setting this parameter to "true", the parent page of a page pop-up is rendered disabled while the pop-up is open. It only applies to page pop-ups. Values: true/false.
reloadpageonbackbutton	Default: false. If set to true, the Ajax framework tries to reload the page when the back button is pressed. A corresponding message box is displayed to inform the end-user about the reload.
requestclienthost	Default: false. If a client sends an HTTP request, it is determined for the first request from which client this request is coming. This operation is sometimes quite expensive. For this reason, you can switch it off. If switched off, there is no disadvantage in normal operation, besides in the monitoring tool you cannot identify which session belongs to which client.

	Values: true/false.
requestdataconverter	Application Designer allows to pass each value that is input by the user through an explicit data converter on the server side, prior to passing this value to the application. In the data converter, you can implement certain security checks, for example, you can prevent users from inputting string sequences containing inline JavaScript or SQL scripting. See the interface com.softwareag.cis.server.IRequestDataConverter for more information. Value: name of a class that implements the interface com.softwareag.cis.server.IRequestDataConverter.
resetstatusbarbefore	Default: false. When set to true, the status bar messages are reset in the browser before a server roundtrip is done. Values: true/false.
sessionidasthreadname	Default: true. On start of each page request processing, the Application Designer runtime calls the method Thread.setName with the current session ID (default). You can set this parameter to "false" to instruct the Application Designer runtime not to touch the thread's name. Values: true/false.
sessiontimeout	Default: 3600 (1 hour). Application Designer sessions are timed out according to the value defined with this parameter. This is the definition of the timeout phase in seconds. By default, 3600 is defined in the configuration file. If no parameter is specified in the configuration file, 7200 is used. Value: integer number.
sdofullpath	Default: false The default setting enables the product's XSLT processor implementation. If switched to true, the 3rd party Xalan implementation is used instead of the product's functionality. Should you decide to use Xalan, download Xalan 2.7.2 from the Apache download sites. **Note:** Xalan 2.7.2 contains a vulnerable.
startmonitoringthread	Default: true. If set to "true", a monitoring thread is opened which by default wakes up every 5 seconds. You can customize this value by setting the

	parameter `monitoringthreadinterval`. The thread performs the following activities: 1. It initiates a garbage collection periodically (every two minutes). 2. It writes all log information into a log file (every *n* milliseconds. Where *n* represents the interval length defined in the `monitoringthreadinterval` parameter). 3. It calls the clean up of sessions which are timed out (every two minutes) 4. It checks for user interface component updates, which need to be deployed. See also *Deploying the User Interface Components*. What happens if the monitoring thread is not started? 1. No garbage collection will be triggered by Application Designer. This is then the task of the servlet container around. 2. The log is not automatically written to the file location specified in the *web.xml* file, but is written to the servlet container's logging. 3. Timing out sessions is not done every two minutes but every thousand requests. 4. No user interface deployment will be done. **Caution:** Some servlet containers do not allow to let the web application start new threads (for example, the Sun reference implementations do so). For these containers, the parameter must be set to "false". Values: true/false.
`suppressfocusmanagement`	Default: false. If you set this parameter to "true", no focus management in the client will be done after a server round trip. This means: The focus will not be set to focus-requesting controls such as "EDIT" fields with "ERROR" status after a server round trip. Usually, you do not set this parameter. If you need to suppress focus management for specific server round trips, you usually do this from within your adapter code for these specific server round trips. See also the `focusmgtprop` in the NATPAGE control. Only set this parameter to "true" if your application needs to do it vice versa: Suppress focus mangement for nearly all server round trips and only explicitly activate focus management for some specific server round trips from within your adapter code. Values: true/false.
`takeoutfieldpopupicon`	Default: false.

	Set this parameter to "true" in case you are using right-aligned FIELD controls with value help. This will avoid overlapping of the right-aligned text and the corresponding drop-down icon. Values: true/false.
`testtoolidhtml4`	Default: false. If set to "true", the HTML attribute generated for the test tool IDs has the name "testtoolid". Otherwise, the name is "data-testtoolid". See also the information on standards mode and HTML5 in the Natural for Ajax documentation.
`textencoding`	Default: UTF-8. By default, Application Designer reads and writes text files in UTF-8 format. You can tell Application Designer to use a different format (for example, for writing XML layout definitions). But be very careful and very aware of what you are doing.
`urlbackbuttonpressed`	When the browser back button is pressed, in some cases the page is not synchronized with the server anymore and the session has to be closed. In these cases a default page is displayed. Instead of this default page you can define a URL to a custom page. Value: the URL of the page that is to be shown instead of the default page.
`urlsessiontimeout`	When Application Designer times out a session (see the `sessiontimeout` parameter) and the user tries to continue to work with the session, a page will be displayed inside the user's browser, indicating that a timeout happened with the user's session. By default, this page is an Application Designer page that you might not want to show to your application users. Value: the URL of the page that is to be shown instead of the default page.
`urlopenstreetmapgeocoder`	URL used to access the third party geocoder for the OPENSTREETMAP controls. You usually do not have to specify this parameter. However, if the URL of the server of this third party geocoder changes, you can adapt the URL here correspondingly.
`uselatestbootstrap`	If set to true Bootstrap 4 is used. If set to false Bootstrap 3 is used. When changing this setting you need to regenerate you page layouts. Default: true. Bootstrap 4 is used. Values: true/false.
`usemessagepopup`	Default: false. Set this parameter to "true" in order to show status messages as message pop-ups.

	Values: true/false.
useownclassloader	Default: true. If set to "true", Application Designer uses its own class loader to load application classes. This parameter may be set to "false" in certain environments, for example, if you use Application Designer inside an environment which requires all application classes to run in the environment's own class loader environment. **Caution:** The Application Designer class loader automatically searches for classes in certain directories (<*project*>/*appclasses*/*classes* and <*project*>/*appclasses*/*lib*). If you do not use the Application Designer class loader, you have to set up your environment accordingly. Values: true/false.
usepagepopup	Default: false. Set this parameter to "true" in order to open Natural for Ajax pop-ups as page pop-ups instead of browser pop-ups. Values: true/false.
valuehelpkeys	You can specify your own keys to open the value help pop-up and/or combo box in a FIELD control. The keys are specified in the same way as hot keys. Example: `valuehelpkeys = "ctrl-65;ctrl-alt-66"`
workplacehotkeys	You can specify hot keys with which you can switch back and forth between the activities in a workplace. The first entry defines the key for forward switching and the second entry defines the key for backward switching. The following example defines CTRL page up and CTRL page down as corresponding hotkeys: `workplacehotkeys = "ctrl-34;ctrl-33"`
xmldatamanager	This parameter defines the file name of the class which implements the `com.softwareag.cis.xmldata.IXMLDataManager` interface. You can specify an own class here. The `com.softwareag.cis.xmldata.XMLDataManagerFactory` creates an instance using a constructor without any parameter.
zipcontent	Default: true. Between the browser and the server, data content is exchanged. By default, Application Designer zips the content before sending a response from the server to the browser client.

	Sometimes you may want to actually “see” what is being sent (maybe you have a test tool that captures the HTTP protocol). Set `zipcontent` to "false" if you do not want Application Designer to zip the data content returned to the client. Values: true/false.

Directory for Performance Traces

The `requestrecording` section of the *cisconfig.xml* file indicates the directory in which recorded performance traces are stored.

```
<cisconfig ...>
    <requestrecording recordrequests="false"
                      recorddirectory="c:/temp/traces/">
    </requestrecording>
</cisconfig>
```

Central Class Path Extensions for Development

If you want to use your own class path extension, you may add a subsection to the *cisconfig.xml* file in which you extend the class path of the Application Designer class loader at development time:

```
<cisconfig ...>
    <classpathextension path="c:/development/centralclasses/classes"/>
    <classpathextension path="c:/development/centralclasses/libs/central.jar"/>
</cisconfig>
```

Each class path extension is listed with a reference to its physical path.

19 Design Time Mode and Runtime Mode

Application Designer may run in two different modes:

Design Time Mode
All resource files which are required by Application Designer are read from the file system using the `cis.home` parameter value inside the ***web.xml*** configuration file.

The Application Designer class loader may be used. This means you can use the feature to dynamically reload application classes without having to restart the web application all the time.

Runtime Mode
All resource files are read internally via mechanisms of the servlet container, by which a web application can access its resource files.

The Application Designer class loader must not be used.

When to Use which Mode

The design time mode is typically used in the following scenarios:

- During development.
- With productive installations, if they are not clustered.

The runtime mode is used in the following scenarios:

- Productive installations which are distributed by the servlet container or application server on several cluster nodes.

The design time mode has the advantage that all resources are read from the file system, and are not blocked after access. This means that you can recreate and change these resources without restarting the web application. This simplifies the development a lot.

Setup

The switch from design time mode and runtime mode is configured in the *web.xml* file:

- If the `cis.home` parameter is set, the design time mode is switched on.

```
<init-param id="CISHOME">
    <param-name>cis.home</param-name>
    <param-value>REALPATH</param-value>
</init-param>
```

- If the `cis.home` parameter is not set, the runtime mode is switched on.

```
<!--
<init-param id="CISHOME">
    <param-name>cis.home</param-name>
    <param-value>REALPATH</param-value>
</init-param>
-->
```

Class Loader Considerations

Application Designer may use its own class loader below the web application's class loader. The purpose of this class loader is to dynamically replace classes during development in order to run newly compiled versions of your software without having to restart the web application all the time.

In the configuration file ***cisconfig.xml***, you can switch this possibility on or off.

See *Appendix D - Class Loader Concepts* for more information on what it means to change between "own Application Designer class loader" and "standard runtime with web application class loader". Both expect classes to be located at different locations. As a consequence, you have to copy classes accordingly in order to bring your application from design time mode to runtime mode.

File Access Considerations

In design time mode (having a defined *cis.home* directory), classes and Application Designer resources (multi-language files) are read from the file system. The reason is that classes can be reloaded without restarting the web application. In runtime mode, this is not done anymore: classes are read by the web application class loader, resources are read via the servlet context.

Consequence: there is no dependency from any file access to a predefined directory - Application Designer is completely clusterable.

20 Natural Web I/O Style Sheets

Name and Location of the Style Sheets

Several aspects on a page (such as font, font style or color) are controlled by a style sheet (CSS file).

The Natural Web I/O Interface client is delivered with the style sheet *3270.css* , which is located in:

../natuniapp.ear/natuniweb.war/resources

Note: For more information on style sheets, see ***http://www.w3.org/Style/CSS/*** .

Editing the Style Sheets

It is recommended that you have a basic understanding of CSS files.

You can edit the predefined style sheets or create your own style sheets.

It is recommended that you work with backup copies. When a problem occurs with your style sheet, you can thus always revert to the original state.

To see your changes in the browser, you have to

1. delete the browser's cache, and
2. restart the session.

Modifying the Position of the Main Output and of the PF Keys

Applies when only the named PF keys are displayed. This feature cannot be used when all PF keys are displayed, since they are always displayed at the same position. See also ***Overview of Session Options*** .

The following elements are available:

Element Name	Description
.mainlayer	Controls the position of the main output in the output window. Used for languages that are written from left-to-right (LTR).
.mainlayer_rtl	Controls the position of the main output in the output window. Used for languages that are written from right-to-left (RTL).
.pfkeydiv	Controls the position of the PF keys in the output window. Used for languages that are written from left-to-right (LTR).
.pfkeydiv_rtl	Controls the position of the PF keys in the output window. Used for languages that are written from right-to-left (RTL).

The *_rtl elements are only used if Natural sends the web I/O screen with a right-to-left flag (SET CONTROL 'VON'). In the browser, the screen elements are then shown on the right side (instead of the left side).

For web I/O in applications where only the left-to-right orientation is used, the *_rtl elements are not required.

If the PF keys are to appear at the bottom, define the elements as shown in the following example:

```
/* Defines the main screen position */ .mainlayer { top: 5px;
     left: 0px;
     height: 550px; } /* Defines the main screen position for right-to-left */ ↩
.mainlayer_rtl{ top: 5px;
     right: 30px;
    height: 550px; } /* Defines the PF keys screen position */ .pfkeydiv { height: ↩
70px;
     left: 0px;
     top: 580px; width: 100%; } /* Defines the PF keys screen position for ↩
right-to-left */ .pfkeydiv_rtl { height: 70px;
     right: 30px;
     top: 580px; width: 100%; }
```

Modifying the Font Size

Depending on the screen resolution, one of the following style sheets for defining the font size is used in addition to the default style sheet:

- *model2.css*
- *model3.css*
- *model4.css*
- *model5.css*

These style sheets are located in the *tmodels* subdirectory of the *resources* directory in which all style sheets are located.

Depending on what comes closest to the standard 3270 screen model, the corresponding style sheet from the *tmodels* subdirectory is automatically used. It is selected according to the following criteria:

Standard 3270 Screen Model	Criteria	Style Sheet
Model 2 (80x24)	30 rows or less.	*model2.css*
Model 3 (80x32)	Between 31 and 40 rows.	*model3.css*
Model 4 (80x43)	41 rows or more.	*model4.css*
Model 5 (132x27)	30 rows or less, and more than 100 columns.	*model5.css*

The font sizes in the above style sheets can be adjusted. Example for *model4.css* :

```
body { font-size: 10px; }
```

The default font sizes for the above 3270 screen models are:

Standard 3270 Screen Model	Default Font Size
Model 2	16px
Model 3	14px
Model 4	10px
Model 5	12px

Modifying the Font Type

As a rule, you should only use monospace fonts such as Courier New or Lucida Console. With these fonts, all characters have the same width. Otherwise, when using variable-width fonts, the output will appear deformed.

If you want to define a different font type, you should define the same font type for the body, the output fields and the input fields as shown in the following example:

```
body {
    background-color: #F3F5F0;
    font-family: Lucida Console;
    }

.OutputField {
    white-space:pre;
    border-width:0;
    font-family: Lucida Console;
```

```
    font-size: 100%;
}

.InputField {
    background-color: white;
    font-family: Lucida Console;
    border-width: 1px;
    font-size: 100%;
    border-color: #A7A9AB;
}
```

Use the CSS at-rule `@font-face` to specify a custom font with which to display text.

For example, Arabic fonts to be used with `InputFields` for the presentation of "Arabic-Indic numerals" instead of "European numerals":

```
@font-face {
font-family: CustomFont;
src: url( CustomFont.woff );
}
```

Defining Underlined and Blinking Text

The following elements are available:

Element Name	Description
`.natTextDecoUnderline`	Defines underlined text.
`.natTextDecoBlinking`	Defines blinking text.
`.natTextDecoNormal`	Defines normal text (no underline, no blinking).

Example:

```
/* Text decoration */
.natTextDecoUnderline { text-decoration:underline; }
.natTextDecoBlinking {text-decoration:blink; }
.natTextDecoNormal {text-decoration:normal;}
```

Blinking text is not supported by the Internet Explorer.

Defining Italic Text

The following elements are available:

Element Name	Description
`.natFontStyleItalic`	Defines italic text.
`.natFontStyleNormal`	Defines normal text (no italics).

Example:

```
/* font style */
.natFontStyleItalic {font-style:italic;}
.natFontStyleNormal {font-style:normal;}
```

Defining Bold Text

The following elements are available:

Element Name	Description
`.natFontWeightBold`	Defines bold text.
`.natFontWeightNormal`	Defines normal text (not bold).

```
/* Font weight */
.natFontWeightBold {font-weight:bolder;}
.natFontWeightNormal {font-weight:normal;}
```

When you define bold text (`{font-weight:bolder;}`) for the default font Courier New, your text always has the same width as with normal text (`{font-weight:normal;}`).

However, when you define bold text for Courier or Lucida Console, the bold text will be wider than the normal text and your output may thus appear deformed. It is therefore recommended that you switch off bold text for Courier and Lucida Console:

```
.natFontWeightBold {font-weight:normal;}
```

Defining Different Styles for Output Fields

The following elements are available:

Element Name	Description
`.FieldVariableBased`	Defines the style for output fields that are based on a variable.
`.FieldLiteralBased`	Defines the style for output fields that are based on a literal.

Example:

```
.FieldVariableBased {
   /* font-style:italic; */
}

.FieldLiteralBased {
    /* font-style:normal; */
}
```

Note: In the above example, as well as in the standard CSS files delivered by Software AG, the variable-based output fields are defined as italic, but are commented out.

Modifying the Natural Windows

The following elements are available:

Element Name	Description
`.naturalwindow`	Controls the rendering of the Natural windows.
`.wintitle`	Controls the rendering of the titles of the Natural windows.

Example:

```
.naturalwindow {
    border-style: solid;
    border-width: 1px;
    border-color: white;
    background-color: black;
}

.wintitle {
    left: 0px;
    top: 1px;
    height: 17px;
```

```
    width: 100%;
    color: black;
    font-size: 100%;
    font-weight: bold;
    background-color: white;
    text-align: center;
    font-family: Verdana;
    border-bottom-style: solid;
    border-bottom-width: 2px;
}
```

Note: In a mainframe environment, you have to set the Natural profile parameter `WEBIO` accordingly to enable this feature. See *WEBIO - Web I/O Interface Screen Rendering* in the *Parameter Reference* which is provided with Natural for Mainframes.

Modifying the Message Line

The rendering of the message line is controlled by the `.MessageLine` element.

Example:

```
.MessageLine {
    color: blue;
}
```

Note: In a mainframe environment, you have to set the Natural profile parameter `WEBIO` accordingly to enable this feature. See *WEBIO - Web I/O Interface Screen Rendering* in the *Parameter Reference* which is provided with Natural for Mainframes.

Modifying the Background Color

The background color is defined in the `body` element.

Example:

```
body {
      background-color: #F3F5F0;
      font-family: Lucida Console;
}
```

Modifying the Color Attributes

You can define different colors for all Natural color attributes. These are:

Red
Green
Blue
Yellow
White
Black
Pink
Turquoise
Transparent

You can define these color attributes for input fields and output fields, and for normal output and reverse video.

The following examples show how to define the color attribute “Red” .

Define the color for a normal output field:

```
.natOutputRed {color: darkred;}
```

Define the foreground and background colors for an output field with reverse video:

```
.reverseOutputRed {background-color: darkred; color:#F3F5F0;}
```

Define the color for a normal input field:

```
.natInputRed {color: darkred;}
```

Define the foreground and background colors for an input field with reverse video:

```
.reverseInputRed {background-color: darkred; color:#F3F5F0;}
```

Modifying the Style of the PF Key Buttons

The following elements are available:

Element Name	Description
`.PFButton`	Controls the style for normal rendering.
`.PFButton:hover`	Controls the style that is used when the mouse hovers over a PF key button.

Example:

```
.PFButton {
    text-align: center;
    width: 90px;
    border-style: ridge;
    border-width: 3px;
    padding: 2px;
    text-decoration: none;
    font-family: Verdana;
    font-size: 12px;
    height: 22px;
}

.PFButton:hover {
    color: #FFFF00;
    background-color: #222222;
}
```

Note: In a mainframe environment, you have to set the Natural profile parameter `WEBIO` accordingly to enable this feature. See *WEBIO - Web I/O Interface Screen Rendering* in the *Parameter Reference* which is provided with Natural for Mainframes.

JavaScript and XSLT Files

In addition to the CSS files described above, the Natural Web I/O Interface client uses XSLT files with specific names for the conversion of the Natural Web I/O Interface screens from the internal XML format to HTML. The HTML also contains calls to JavaScript. For Web I/O screens the following conversion is done:

- Input text is placed into the HTML element `<input>` .
- Output text is placed into the HTML element `<input>` (with attribute `readonly="readonly"`).
- A message line is placed into the HTML element `<span>` .
- PF keys are embedded in an XML island and then rendered with JavaScript.
- Window elements are embedded in an XML island and then rendered with JavaScript.

The conversion is done at runtime by the following product files of your web application:

1. *<mywebapp>/WEB-INF/***transuni.xsl**
2. *<mywebapp>/scripts/***natuniscript.js**

The XSLT file *transuni.xsl* is only read once when the server is started.

Important: Do not change the above files. Software AG may change the functionality of these files in new versions or service packs of the product, which would overwrite your changes.

Customizing JavaScript

You can copy your own JavaScript file with extended JavaScript functionality into the directory *<mywebapp>/scripts*. This file must have the name ***usernatunicscript.js***. Define your new functionality in this JavaScript file.

If a *usernatunicscript.js* is found when the server is started, it is read additionally to the JavaScript file *natunicscript.js*.

For the new JavaScript functionality to be executed, you may also need to **customize the XSLT file**.

Customizing XSLT

Make a copy of *<mywebapp>/WEB-INF/transuni.xsl* and save it as *<mywebapp>/WEB-INF/**usertransuni.xsl***. Modify the XSL elements to your needs.

After making changes to *usertransuni.xsl* user file, you have to restart the server so that your changes become effective. If a *usertransuni.xsl* file is found when the server is started, it is read instead of the default XSLT file.

Customizing Overwrite / Insert mode in Input fields

Up to NJX 9.1.1, Internet Explorer always used “overwrite” mode whereas all other browsers used “insert” mode. With NJX 9.1.2 this inconsistency was fixed. The default behavior is now “overwrite” mode for all browsers. This reflects the default behavior of Natural applications.

If you prefer "insert" mode, do the following:

1. Copy the file *WEB-INF/transuni.xsl* to *WEB-INF/usertransuni.xsl* – refer to the section **customize the XSLT file** above.
2. Open *usertransuni.xsl* in a text editor. Search for the following line:

   <xsl:attribute name=”onkeypress”>handleKeyPress(this.name, this);</xsl:attribute>
3. Remove this line and save the file.

Packaging Customized Files in Natural for Ajax WAR Files

When using the ***Deployment Wizard for Web Applications*** of NaturalONE to deploy your Natural for Ajax *.war* file, you can automatically package a custom *usertransuni.xsl* in the *.war* file for deployment:

- Add a webconfig directory to your NaturalONE project as described in the documentation *NaturalONE > Natural for Ajax > Deploying the Application > **Content of the Sample webconfig Directory***.
- Copy your *usertransuni.xsl* to the *webconfig/web-inf* folder of your NaturalONE project.

21 Multi-Language Management

The multi-language management is responsible for changing the text IDs into strings that are presented to the user.

There are two translation aspects:

- All literals in the GUI definitions of a layout are replaced by strings which are language-specific.
- Literals you output within your adapter code (e.g. status messages) must be translated.

The multi-language management is internally kept cleanly behind an internal interface. This means that in the future a different implementation will be available to provide a solution to find a string for a given text ID. In this , the default implementation which simply uses comma separated value files is described.

The information provided in this is organized under the following headings:

Writing Multi-Language Layouts

When defining properties of controls inside a layout definition, there are always two options to specify fix labels: either use property `name` or property `textid`. In case your pages support multi-language ability, you only have to use the `textid` property. At runtime, the corresponding labels are found in the following way:

- Each PAGE has the property `translationreference`. This property may be the name of the HTML file - or it may be a logical name, used by different HTML pages.
- Inside Application Designer, there are defined directories and files in which the text information is stored: each application project is represented by a directory under the web application directory of Application Designer. Inside the project directory, there is a directory */multilanguage/*. Under this directory, each language is represented by its own directory, e.g. by the directory */multilanguage/de/* for German translations.
- Inside each language directory, there is one comma separated value (CSV) file for each page name. The name of the file is *<pagename>.csv* (for example, *Login.csv*).
- Inside the CSV file, each line contains the text ID, a semicolon and the label text, e.g. "Label1;Login name".

    - Example

- Page Name Strategy

Example

Let us assume you have defined an application project "accountmgmt". Inside the application project, there is a layout definition *account.xml* that points via the `translationreference` property of PAGE to "account". The file structure inside your application project directory now looks as follows:

```
<webapp-directory>/
  accountmgmt/
    account.html                        // generated HTML file
    multilanguage/
      de/
        account.csv                     // German text
      en/
        account.csv                     // English text
    xml/
       account.xml                      // layout definition
```

Page Name Strategy

The previous section explained how a translation file is found for a certain HTML page. Basically, the translation reference is used to link the layout definition and the Application Designer multi-language management.

In general, there are two strategies for using this translation reference, and a mixture of both:

- Specify one central page name for a couple of pages. Therefore, all pages share the same multi-language information (i.e. the same *.csv* file).
- Specify one page name for each page. Therefore, every page has its own *.csv* file.

For larger projects, it makes sense to combine different literal information into one file - in order to keep consistency and to avoid redundancy. Of course, you have to synchronize the naming of text IDs for each page.

Creating the Translation File

The translation file (*account.csv* in the **example** of the previous section) is a simple comma separated file with the following format:

```
textid1;text1
textid2;text2
textid3,text3
```

If your text itself contains a semicolon, then write "\;".

You can either create the file by using a text editor or you can use Application Designer's Literal Assistant which is integrated in the Layout Painter.

Pay attention: when using text editors of your own, you must configure your editor to store the text using UTF-8 character encoding. Otherwise, any characters that are not "ASCII characters < 128" will not be properly displayed. Make sure that your editor is UTF-8 capable.

Tools for Translating Text IDs

There are two tools. One is the Literal Assistant that is part of the Layout Painter. The other is the Literal Translator.

Tool for Creating Languages

Application Designer comes with two languages: "en" for English and "de" for German. When creating a new language abbreviation, you have to take care of the following:

- You have to create language directories in your projects.
- You have to copy certain files in which Application Designer holds text information that is language dependent.

The Language Manager automates the creation of language abbreviations.

Unicode

Pay attention to the fact that Application Designer is fully based on Unicode and its UTF-8 format. All multi-language files must be in UTF-8 format. Especially pay attention when maintaining CSV files with programs like MS Excel.

22 Starting a Natural Application with a URL

The connection parameters available in the configuration file for the session and on the logon page can also be specified as URL parameters of the logon page URL. This allows bookmarking the startup URL of a Natural application or starting an application by clicking a hyperlink in a document.

The URL parameters overrule the definitions in the configuration file, with the exception described in the table below.

The following URL parameters are available for the logon page:

URL Parameter	Corresponding Option in the Session Configuration
`natsession`	**Session ID**
`natserver`	**Host name**
`natport`	**Port number**
`natuser`	**User name**
`natprog`	**Application**
`natparam`	**Natural parameters**
`natparamext`	**Natural parameters** The URL parameter `natparamext` extends an existing Natural parameter definition in the configuration file. The extension works in the following way: the Natural parameters defined in the configuration file come first. Then, the Natural parameters defined in the URL parameter `natparamext` are added, separated by a space character. If you want to overrule the definition in the configuration file, use the URL parameter `natparam` instead.
`nattimeout`	**Timeout (n seconds)**

Important: All parameter values must be URL-encoded.

Example: In order to start the Natural program `dump`, while your application server is running on *myhost:8080* and your Natural Web I/O Interface server is running on *myserver1:4811*, you can use the following URL:

http://myhost:8080/natuniweb/natural.jsp?natserver=myserver1&natport=4811&natprog=dump&natuser=myusername

23 Configuring Container-Managed Security

General Information

The Natural Web I/O Interface client comes as a Java EE-based application. For the ease of installation, the access to this application is by default not secured. You might, however, wish to restrict the access to certain parts of the application to certain users. An important example is the **configuration tool** , which enables you to modify the Natural session definitions and the logging configuration of the Natural Web I/O Interface client . Another example is the Natural logon page.

This section does not cover the concepts of JAAS-based security in full extent. It provides, however, sufficient information to activate the preconfigured security settings of the Natural Web I/O Interface client and to adapt them to your requirements.

Name and Location of the Configuration File

Security is configured in the file *web.xml* . This file is located in the following directory:

<tomcat-install-dir>/webapps/natuniweb/WEB-INF

Activating Security

Great care must be taken when editing and changing the configuration file *web.xml* . After a change, the application server must be restarted.

Edit the file *web.xml* and look for the section that is commented with "Uncomment the next lines to add security constraints and roles." . Uncomment this section by removing the comment marks shown in boldface below:

```
<!-- Uncomment the next lines to add security constraints and roles. -->
<!--
<security-constraint>
    <web-resource-collection>
    <web-resource-name>Configuration Tool</web-resource-name>
        <url-pattern>/conf_index.jsp</url-pattern>
        <url-pattern>/faces/*</url-pattern>
    </web-resource-collection>
...
<security-role>
    <description>Administrator</description>
    <role-name>nwoadmin</role-name>
</security-role>
-->
```

Defining Security Constraints

The security constraints defined by default are just examples. A `<security-constraint>` element contains of a number of `<web-resource-collection>` elements combined with an `<auth-constraint>` element. The `<auth-constraint>` element contains a `<role-name>` . The whole `<security-constraint>` element describes which roles have access to the specified resources.

Example - the following definition specifies that only users in the role "nwoadmin" have access to the configuration tool:

```
<security-constraint>
    <web-resource-collection>
    <web-resource-name>Configuration Tool</web-resource-name>
        <url-pattern>/conf_index.jsp</url-pattern>
        <url-pattern>/faces/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>nwoadmin</role-name>
    </auth-constraint>
</security-constraint>
```

In the following section, you will see where and how the roles are defined.

Defining Roles

A few lines below in the file *web.xml* , there is a section `<security-role>` . Here, the roles that can be used in `<security-constraint>` elements are defined. You can define additional roles as needed. The assignment of users to roles is done outside this file and will often be done in a user management that is already established at your site.

Example:

```
<security-role>
    <description>Administrator</description>
    <role-name>nwoadmin</role-name>
</security-role>
```

Selecting the Authentication Method

In the file *web.xml* , there is a section `<login-config>` . The only element that should possibly be adapted here is `<auth-method>` . You can choose between the authentication methods "FORM" and "BASIC" . Form-based authentication displays a specific page on which users who try to access a restricted resource can authenticate themselves. Basic authentication advises the web browser to retrieve the user credentials with its own dialog box.

Example:

```
<login-config>
    <auth-method>FORM</auth-method>
...
</login-config>
```

Configuring the UserDatabaseRealm

In the *tomcat-users.xml* file (which is located in the *conf* directory), specify the role "nwoadmin" for any desired user name and password. For example:

```
<user username="pepe" password="pepe123" roles="nwoadmin"/>
```

For detailed information on the necessary realm configuration for Tomcat, see ***http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html#UserDatabaseRealm*** .

24 Configuring SSL

General Information

Trust files are used for a secure connection between the Natural Web I/O Interface server and the Natural Web I/O Interface client . Server authentication cannot be switched off. A trust file is always required.

A trust file contains the certificates that you trust. These can be certificates of a CA (Certificate Authority) such as VeriSign, or self-signed certificates.

For information on the steps that are required on the Natural Web I/O Interface server and how to generate a self-signed certificate which needs to be imported to the client, see *SSL Support* .

To establish a secure connection, you have to proceed as described in the topics below.

Creating Your Own Trust File

To create your own trust file, you can use, for example, Sun's keytool utility which can be found in the *bin* directory of the Java Runtime Environment (JRE). Here are some helpful examples:

- Create an empty, password-protected trust file:

```
keytool -genkey -alias foo -keystore truststore.jks -storepass "your-passwort"
keytool -delete -alias foo -keystore truststore.jks
```

- Import a certificate:

```
keytool -import -alias "name-for-ca" -keystore truststore.jks -storepass ↩
"your-passwort" -file server.cert.crt
```

  You should use a meaningful name for the alias.

- List the certificates in a trust file:

```
keytool -list -v -keystore truststore.jks
```

- Delete a certificate from a trust file:

```
keytool -delete -alias "name-for-ca" -keystore truststore.jks
```

When you modify the trust file or its password, you have to restart the application server so that your modification takes effect.

Defining SSL Usage in the Configuration File

Invoke the **configuration tool** and proceed as follows:

1. In the global settings for all defined sessions, define the **SSL trust file path** and, if required, the **SSL trust file password** . See also *Global Settings* in *Natural Client Configuration Tool* .

   With the server authentication, the Natural Web I/O Interface client checks whether the certificate of the Natural Web I/O Interface server is known. If it is not known, the connection is rejected.

   When a trust file is not defined in the configuration tool, the Natural Web I/O Interface client tries to read the file *calist* from the *lib/security* directory of the Java Runtime Environment (JRE). The default password for this file is "changeit" .
2. Define a session and set the session option **Use SSL** to **Yes** . See also *Overview of Session Options* in *Natural Client Configuration Tool* .

25 Logging

General Information

The Natural Web I/O Interface client uses the Java Logging API. In case of problems with the Natural Web I/O Interface client , you can enable logging and thus write the logging information to an output file. This should only be done when requested by Software AG support.

You configure logging using the **configuration tool** .

Note: Some logging information is also written to the console, regardless of the settings in the configuration file. The console shows the information which is normally provided by the logging levels `SEVERE` , `WARNING` and `INFO` .

Name and Location of the Configuration File

The name of the configuration file is *natlogger.xml* , which is located in: .

`<application-server-install-dir>` *server/default/deploy/naturalunicode.rar/log*

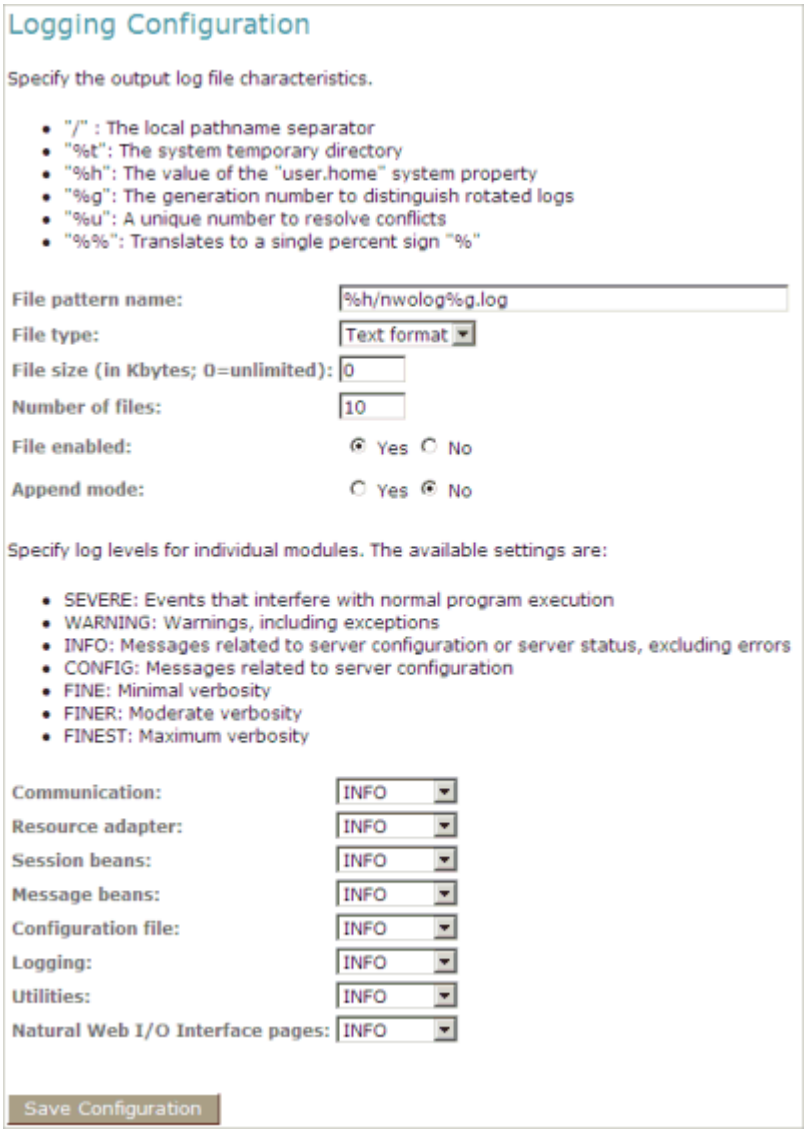
Invoking the Logging Configuration Page

The content of the configuration file *natlogger.xml* is managed using the **Logging Configuration** page of the **configuration tool** .

To invoke the Logging Configuration page

1 In the frame on the left, choose the **Logging Configuration** link.

The **Logging Configuration** page appears in the right frame. Example:



2 Specify the characteristics of the output file as described below in the section *Overview of Options for the Output File* .

3 Specify the log levels for individual modules by selecting the log level from the corresponding drop-down list box.

A brief description for each log level is provided on the **Logging Configuration** page.

4 Choose the **Save Configuration** button to write the modifications to the configuration file.

Caution: When you do not choose the **Save Configuration** button but log out instead or leave the configuration tool by entering another URL, your modifications are not written to the configuration file.

Overview of Options for the Output File

The following options are provided for specifying the characteristics of the output file:

Option	Description
File pattern name	The pattern for generating the output file name. Default: "%h/nwolog%g.log" . The default value means that an output file with the name *nwolog <number>.log* will be created in the home directory of the user who has started the application server. For detailed information on how to specify the pattern, see the Java API documentation .
File type	The format of the output file. Select one of the following entries from the drop-down list box: ■ **Text format** Output in simple text format (default). ■ **XML format** Output in XML format. The corresponding formatter class is then used.
File size	The maximum number of bytes that is to be written to an output file. Zero (0) means that there is no limit. Default: "0" .
Number of files	The number of output files to be used. This value must be at least "1" . Default: "10" .
File enabled	If set to **Yes** (default), the file handler is enabled. If set to **No** , the file handler is disabled.
Append mode	If set to **Yes** , the logging information is appended to the existing output file. If set to **No** (default), the logging information is written to a new output file.

VII Operating and Monitoring the Natural Web I/O Interface Server

This part covers the following topics:

Notation *vrs* or *vr*

When used in this documentation, the notation *vrs* or *vr* represents the relevant product version (see also *Version* in the *Glossary*).

26 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.

Starting the Natural Web I/O Interface Server

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* represents the relevant product version 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 Linux System Services.

Note: If you qualify the Natural Web I/O Interface server data sets 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'
```

Terminating the Natural Web I/O Interface Server

The Natural Web I/O Interface server can be terminated from within the Monitor Client `NATMOPI`, see ***Monitor Commands*** below.

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.

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*.

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 31+30+15`
`list sessions`	Returns a list of active Natural sessions within the server. For each session, the server returns information about the user who owns the session, the session initialization time, the last activity time and an internal session identifier (*`session-id`*).
`cancel session` *`session-id`*	Cancels a specific Natural session within the Natural Web I/O Interface server. To obtain the session ID, use the monitor command `list sessions`. Canceling active batch server sessions in Natural requires authorized services provided by the Authorized Services Manager (ASM). If the ASM is not started, those sessions cannot be canceled.
`cleanup`	Cancel sessions that are inactive longer than specified in configuration parameter `SESSION_TIMEOUT`.
`help`	Returns help information about the monitor commands supported.

Runtime Trace Facility

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

The following topics are covered below:

- Trace Medium
- Trace Configuration

- Trace Level

Trace Medium

The Natural Web I/O Interface 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 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 22-17	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 11-01	Free.
Bit 00	Reserved for trace-level extension.

Note: Using trace levels 12 and 13 is only possible with Natural Web I/O Interface Server Version 8.3.2.

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.

27 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.

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 Web I/O Interface 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

Example: Ping a Server in Different Environments

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

- System operator:

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

- Com-plete online command:

```
NATMOPI -sServerName -cPING
```

Server in z/OS (started task or batch mode):

- Execute `NATMOPI` in batch job:

```
NATMOPI,PARM=('-sServerName -cPING')
```

Sample job:

```
//SAGMOPI  JOB  SAG,CLASS=K,MSGCLASS=X
//NATEX EXEC PGM=NATMOPI,REGION=3000K,
// PARM=('-Sname -CPING')
//* PARM=('-H')
//STEPLIB  DD DISP=SHR,DSN=NATURAL.XXXvr.LE.LOAD
//         DD DISP=SHR,DSN=CEE.SCEERUN
//SYSOUT   DD   SYSOUT=X
//SYSPRINT DD   SYSOUT=X
//*
```

Where *XXX* is the Web I/O Interface server product code (NWO) and *vr* is the two-digit product version.

- Execute `NATMOPI` in TSO (Command):

```
NATMOPI -sServerName -cPING
```

The NWO load library must be included in the steplib of TSO.

Further Command Examples:

`natmopi -dls`	List all servers registered in the directory in short format.
`natmopi -dcl TST -ls TST`	Clean up all servers with ID `TST*` (ping server and remove it, if it does not respond), and list all servers with ID `TST*` after cleanup.
`natmopi -sSRV1 -cping -sSRV2 ↩ -sSRV3 -cterminate`	Send command `ping` to `SRV1`. Send command `terminate` to `SRV2` and `SRV3`.

`natmopi -cterminate -sSRV1 ↩ -cping -sSRV2 -sSRV3`	Is equivalent to the previous example. That is, `NATMOPI` sends the command following the `-s` option to the server. If no `-c` option follows the `-s` option, the first `-c` option from the command line will be used.
`natmopi -sSRV1 -cterminate -a`	Send command `terminate` to `SRV1`. If `SRV1` is ambiguous in the server directory, send the command to all `SRV1` servers without prompting for selection.

28 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 commands 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.

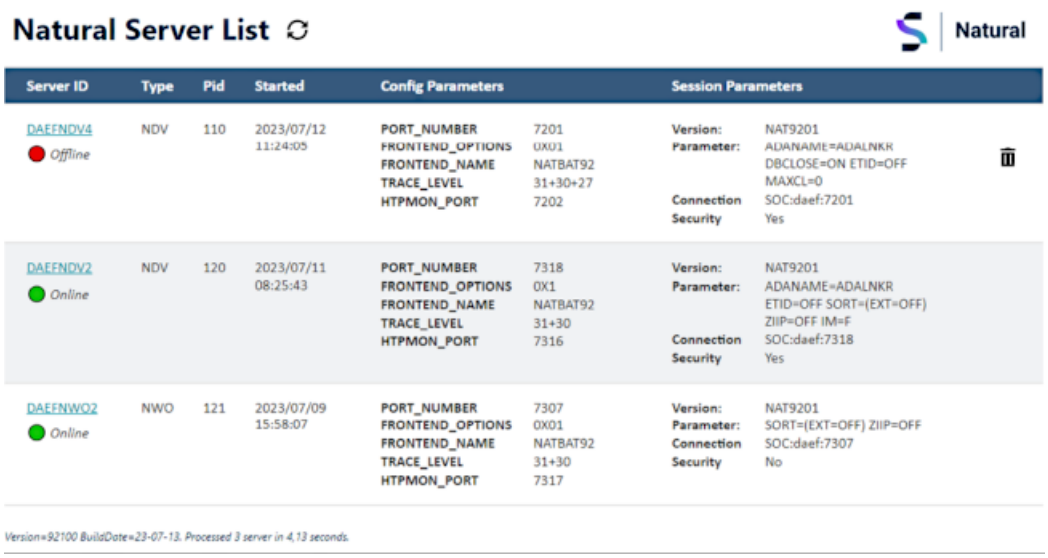
Prerequisites for HTML Monitor Client

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 Web I/O Interface server address space. It is configured with the NWO server 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.

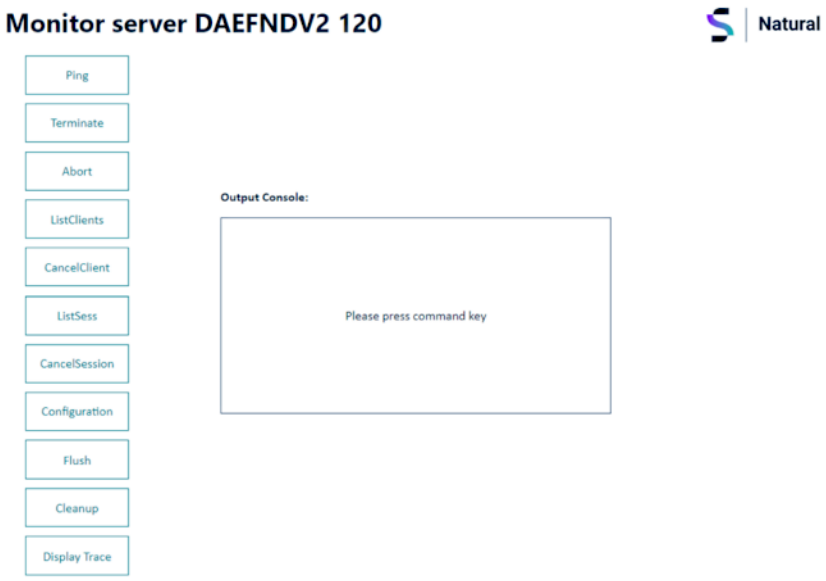
Example



The server list consists of online and offline servers. The offline servers represent potentially dead server entries which can be deleted from the server directory by choosing the **Remove** icon. **Remove** appears only for offline servers. “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 offline, it still might be active but could not respond to the ping. Choosing **Remove** does not terminate such a server but removes its reference in the monitor directory. Hence, it cannot be reached by the monitor anymore.

Choosing a link of the server ID opens a window for monitoring the selected server. Clicking the link with the left mouse button opens the server in a popup window. Clicking the link with the right mouse button opens the server monitor either in a new tab or in a new browser window, depending on the selection in the context menu.

Server Monitor



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 because it is a sample value.

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

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

ListSess shows currently connected sessions. Connected sessions are either permanent connections to Natural Studio or temporary connections to NaturalONE. In case of NaturalONE, a connected session is a session that currently executes a Natural session (for example using **Run As/Debug As**), waits on an I/O screen or uses the **Natural Command Console for Mainframes**.

In conjunction with Security Caching (see parameter *SECURITY_CACHING* in the documentation of Natural Development Server) the following buttons are available:

ListClients displays the clients that are logged in, including the following information (from left to right):

- Number of client entry
- UserId - UserId of the client
- Host - IP Address of the client
- LogonTime - Time of first login of the client
- LastActivity - Time of last access of the client
- St - Status of the client:

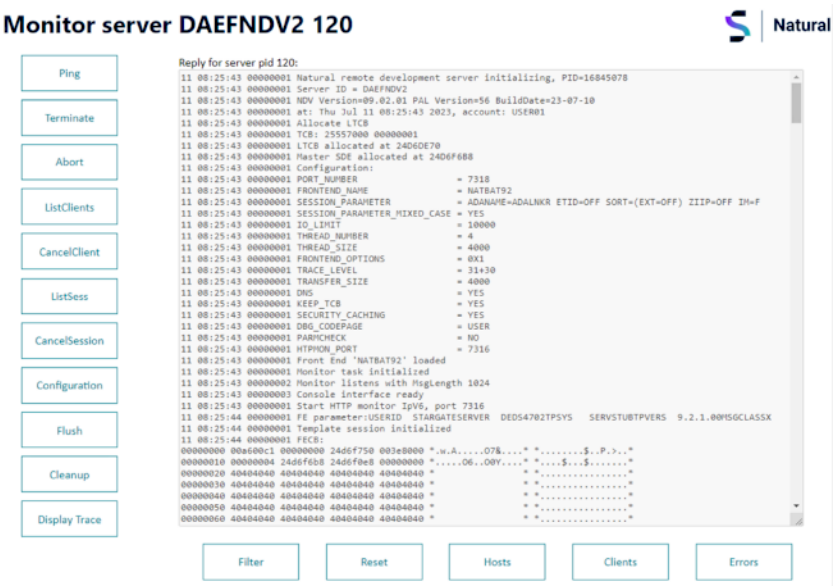
  A - Client successfully validated by RACF
  N - Client not validated by RACF

CancelClient deletes a client and forces a new password prompt. To delete a client from the list, select the number assigned to the client and choose **CancelClient**, or enter the number of the client after choosing **CancelClient**.

Display Trace provides basic filter functions for the trace output of a server. The trace output is loaded completely into a HTML page. In case of very large trace files, loading the files might load with a delay. To refresh the content of the trace, for example if new lines were written, press **Display Trace** again.

Each trace line consists of day number, time, thread ID, and message text. The thread ID is an increasing number starting with 0 or 1 (main thread). Each new thread gets a new ID. The main thread starts several new threads like HTTP Monitor, Console Interface, or Main Listener. Each request from a client (like NaturalOne) creates a new thread. Therefore, it is possible to filter particular client sessions.

You can also use well-known browser functions, such as cut and paste or find, to navigate inside a trace.



- Filter

  Enter a text to filter the trace. Only lines containing the filter text are displayed.

- Reset

  Filter value is being reset. The entire loaded trace is displayed again.

- Hosts

  Predefined filter that shows the IP address and, if known, the name of all hosts that had logged into the Server. You can now, for example, display the progress of a particular session by selecting a thread ID and clicking the **Filter** button.

- Clients

  Predefined filter that shows the name of clients that had logged into the Server. As in the previous function, you can now, for example, display the progress of a particular session by selecting a thread ID and clicking the **Filter** button.

- Errors

Predefined filter that shows errors that occur on the Server. You can now, for example, display the progress of a particular session by selecting a thread ID and clicking the **Filter** button.

Example:

Filter trace for clients, select the thread ID of a particular client session, and click **Filter**. You will see all trace lines of this particular client session.

Index