software AG
A SOFTWARE GMBH BRAND

Natural

Natural Web I/O Interface

Version 9.3.1

September 2025

ADABAS & NATURAL

Document ID: NATUX-NNATWEBIO-931-20250930

Table of Contents

Preface

This documentation is organized under the following headings:

Introduction	What is the Natural Web I/O Interface?
Installing and Configuring the Natural Web I/O Interface Server	How to install and configure the Natural Web I/O Interface server in a Linux 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.

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

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

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.

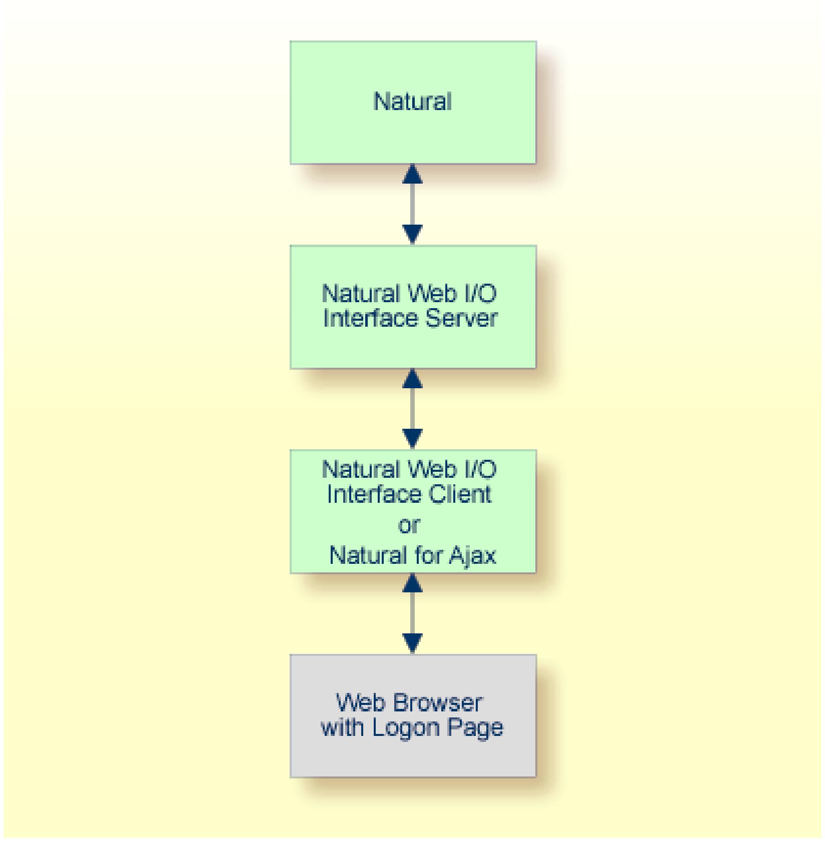
Two types of client are supported:

- 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:


Natural
Natural Web I/O Interface Server
Natural Web I/O Interface Client or Natural for Ajax
Web Browser with Logon Page

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 Installing and Configuring the Natural Web I/O Interface Server

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

On Linux, the server part of the Natural Web I/O Interface runs in the background as a so-called daemon.

Installing the Natural Web I/O Interface Daemon

The Natural Web I/O Interface daemon is installed with Natural for Linux if the corresponding option is set during the installation. See the *Installation* documentation for further information.

Setting Up the Natural Web I/O Interface Components

Setting up the Natural Web I/O Interface on Linux consists of the following steps:

- Step 1: Stop the Natural Web I/O Interface Daemons
- Step 2: Establish the Environment
- Step 3: Install Natural and the Natural Web I/O Interface
- Step 4: Check the Environment Variables for the Natural Web I/O Interface
- Step 5: Read the READ_NWO Files

Step 1: Stop the Natural Web I/O Interface Daemons

This step is only required for an upgrade installation. It is not required when you install the Natural Web I/O Interface for the first time.

1. Stop the *nwosrvd* process using the following command:

   ```
   nwosrvd.sh portnumber stop
   ```

   Or use the script *$NAT_HOME/INSTALL/nwosrvd.bsh* which will be generated during the Natural Web I/O Interface installation for a specified port.

   ```
   nwosrvd.bsh stop
   ```

2. Repeat the above command (with an adapted port in script *nwosrvd.bsh*, if applicable) for each Natural Web I/O Interface service that is needed.

Step 2: Establish the Environment

- Besides the settings for the Natural environment, additional settings for the Natural Web I/O Interface environment must be set. Ensure that the environment settings for Natural are set by the *natenv* environment script. The *nwoenv* environment script is called by the *natenv* environment script. Therefore, the Natural Web I/O Interface environment will be set with the Natural environment if it is set after the Natural Web I/O Interface installation.

  If the Natural Web I/O Interface environment is to be separate from the Natural environment, use the shell script *nwoenv* or *nwoenv.csh* by entering one of the following commands:

  ```
  source nwoenv
  ```

  ```
  source nwoenv.csh
  ```

  These scripts can be found after the installation in *$NAT_HOME/INSTALL*.

Step 3: Install Natural and the Natural Web I/O Interface

- The Natural Web I/O Interface can be selected in the **Choose Packages** screen during the Natural installation.

  Optionally, you may install a runlevel script to start/stop a Natural Web I/O Interface daemon when the machine starts/stops.

  After the Natural installation has finished, the Natural Web I/O Interface must be activated by starting Natural through a Natural Web I/O Interface client on Windows.

  When a runlevel script is used, the Natural Web I/O Interface daemon can only be administered by the user "root".

  When you install Natural with the Natural Web I/O Interface, the directory *$NAT_HOME/nwo/$NWONODE* is created. The template files located in *$NAT_HOME/nwo/node-name* are then copied to this new directory.

Step 4: Check the Environment Variables for the Natural Web I/O Interface

- The Natural Web I/O Interface-specific settings are shown below:

Environment Variable	Description
`NWODIR`	The home directory for the product located at *$NAT_HOME/nwo*.
`NWONODE`	The name of the node (machine) on which the Natural Web I/O Interface is installed.
`NWO_SRVDCONF`	The configuration file *$NAT_HOME/nwo/$NWONODE/nwosrvd.conf* for the Natural Web I/O Interface daemon.

Environment Variable	Description
`NWO_TIMEOUT`	The maximum time, in seconds, that the Natural Web I/O Interface daemon will wait for a response. "0" means no timeout. The Natural Web I/O Interface daemon will terminate when it receives the timeout.

Step 5: Read the READ_NWO Files

1. Access the directory *$NAT_HOME/nwo* and check the files *READ_NWO.TXT* and *READ_NWO.FIX* for any version-specific installation considerations concerning the particular platform.
2. Add the services as described in the file *READ_NWO.TXT*.

Directories

The following directories are created when Natural is installed together with the Natural Web I/O Interface on a Linux system:

Directory	Description
$NAT_HOME	Top-level Natural directory.
$NATDIR	Only used for compatibility with previous versions. Top-level Natural directory.
$NATDIR/$NATVERS	Only used for compatibility with previous versions. The version subdirectory has been removed. Since *$NATVERS* is set to ".", it equals *$NAT_HOME* and *$NATDIR*.
$NWODIR	Directory with the Natural Web I/O Interface components for the current version.
$NWONODE	Contains the name of the machine (`uname -n`).
$NAT_HOME/INSTALL	Shell scripts and environment files for the Natural Web I/O Interface (*nwoenv*, *nwoenv.csh*).
$NWODIR/bin	Natural Web I/O Interface executable files (*nwosrvd*, *nwosrvd.tr*).
$NWODIR/node-name	Contains the template files (*nwosrvd.sh*, *nwo.sh*, *nwosrvd.conf*).
$NWODIR/nwoexuex/userexit1	Contains the files for building the *libnwouserexit1*.
$NWODIR/nwoexuex/userexit2	Contains the files for building the *libnwouserexit2*.
$NAT_HOME/nwo/$NWONODE	Work directory, contains the configuration files (*nwosrvd.sh*, *nwo.sh*, *nwosrvd.conf)*.

Note: The above table lists the most important directories and files.

Configuring the Natural Web I/O Interface Daemon on Linux

When the Natural installation has finished, the directory *$NAT_HOME/nwo/$NWONODE* contains the files *nwosrvd.conf*, *nwosrvd.sh* and *nwo.sh*.

The configuration of the Natural Web I/O Interface daemon can be done using the Natural Web I/O Interface daemon commands or by editing the configuration file *nwosrvd.conf*.

The following topics are covered below:

Natural Web I/O Interface Daemon Commands

The following commands can be specified at the Linux command prompt:

Command	Description
`nwosrvd -help`	Shows all available Natural Web I/O Interface daemon commands and subcommands.
`nwosrvd -v`	Shows the version of the Natural Web I/O Interface daemon.
`nwosrvd` *nnnn*	Defines the listening port number.
`nwosrvd -show`	Shows the configuration of the Natural Web I/O Interface daemon.
`nwosrvd -config` *keys*	Changes the configuration of the Natural Web I/O Interface daemon. The following keys can be specified: `-host=`*hostname* The host name used. `-userexit1=`*pathname* The message defined with this key is saved in the `UserExit1` key of the configuration file *nwosrvd.conf*, section `[UserExits]`. `-userexit2=`*pathname* The message defined with this key is saved in the `UserExit2` key of the configuration file nwosrvd.conf, section `[UserExits]`. `-passparam=`*parameters* The message defined with this key is saved in the `Parameters` key of the configuration file *nwosrvd.conf*, section `[PasswdArguments]`. `-passold=`*message* The message defined with this key is saved in the `EnterOldPassword` key of the configuration file *nwosrvd.conf*, section `[PasswdMessages]`.

Command	Description

	`-passnew=`*`message`*
	The message defined with this key is saved in the `NewPassword` key of the configuration file *nwosrvd.conf*, section `[PasswdMessages]`.
	`-passreenter=`*`message`*
	The message defined with this key is saved in the `ReEnterNewPassword` key of the configuration file *nwosrvd.conf*, section `[PasswdMessages]`.
	`-passsuccess=`*`message`*
	The message defined with this key is saved in the `PasswordSuccessful` key of the configuration file *nwosrvd.conf*, section `[PasswdMessages]`.
	`-logging=`*`option`*
	The option defined with this key is saved in the `Logging` key of the configuration file *nwosrvd.conf*, section `[Miscellaneous]`.
	`-ssl=[yes|no]`
	The option defined with this key is saved in the `ssl` key of the configuration file *nwosrvd.conf*, section `[SSL]`.
	`-pam=[yes|no]`
	The option defined with this key is saved in the `pam` key of the configuration file *nwosrvd.conf*, section `[PAM]`. PAM itself also has a configuration file or section (depends on the PAM implementation); the PAM configuration name must be `nwosrvd`.
	To remove any user exits from the configuration, enter the following command:
	`nwosrvd -config -userexit1=`
	Once the configuration was changed, the Natural Web I/O Interface daemon must be restarted.

nwosrvd.conf - Configuration File for the Natural Web I/O Interface Daemon

The configuration file *nwosrvd.conf* contains information that the user exits need for the Natural Web I/O Interface daemon. It has the following content:

```
[Miscellaneous]
Logging=I

[UserExits]
; UserExit1=/FS/sag/nat/nwoexuex/userexit1/libnwouserexit1.so
; UserExit2=/FS/sag/nat/nwoexuex/userexit2/libnwouserexit2.so

[PasswdArguments]
Parameters=

[PasswdMessages]
EnterOldPassword=Enter existing login password:
NewPassword=New Password:
```

```
ReEnterNewPassword=Re-enter new Password:
PasswordSuccessful=passwd: password successfully changed for*

[SSL]
ssl=no

[PAM]
pam=no

[HA]
ha=no
```

Section in Configuration File	Description
`[Miscellaneous]`	The key `Logging` is used to define the amount of logging information that is to be reported. One of the following options can be specified: `E` for errors. `W` for warnings. `I` for information. See also ***Logging Information***.
`[Host]`	The hostname used. (optional)

`[UserExits]`	Two user exits can be defined:
	
	`UserExit1`
	
	The library that is defined by `UserExit1` contains the following function:
	
	`int nwo_CheckUsernameAndPassword(const char *pUsername, const char *pPassword, const char *pNewPassword, char *pErrorMessage)`
	
	If the key `UserExit1` is defined in the configuration file, the function `nwo_CheckUsernameAndPassword` is responsible for checking the user name and password. If a new password is received, user exit 1 is also responsible for changing the password.
	
	In the case of an error, the return code of the function must be "0"; in this case, the `pErrorMessage` is returned to the client.
	
	When user name and password are correct, the return code must be a value other than "0". "1" indicates that the Natural session runs under the user who started the daemon (authentication). "2" indicates that the Natural session runs under the login user (authentication and impersonation).
	
	`UserExit2`
	
	The library that is defined by `UserExit2` contains the following functions:
	
	■ `int nwo_Messages(int *iNumberOfMessages, char *pMessage[])`

Section in Configuration File	Description
	`iNumberOfMessages`: Number of messages returned in the array. `pMessage`: Array of messages. If the key `UserExit2` is defined in the configuration file, the function `nwo_Messages` is called when a new connection (client) is accepted and the messages returned by this function are sent to the client. User exit 2 may be used, for example, to send a message such as the following: “For maintenance reasons, the Natural application XXXXX will be down next monday, from 18:00 until 19:00”. In the case of an error, the return code of the function must be "0". After the function `nwo_Messages` has been called, the function `nwo_FreeMessages` is called. ■ `int nwo_FreeMessages(int iNumberOfMessages, char *pMessage[])` `iNumberOfMessages`: Number of messages. `pMessage`: Array of messages. If the key `UserExit2` is defined, the function `nwo_FreeMessages` is called to free any resources (normally memory) allocated in the function `nwo_Messages`. In the case of an error, the return code of the function must be "0".
`[PasswdArguments]`	The key `Parameters` is used to define any additional parameter(s) that have to be passed to the `passwd` command.
`[PasswdMessages]`	The keys in this section define the messages that are to be returned by the system (`passwd` command) when a user changes the password. If any of these messages is not identified by the daemon, an error will be returned to the client. **Password Mechanism** The password and new password are encrypted on the client side and decrypted on the Linux side. A maximum of 8 characters is allowed. If user exit 1 is active, user name, password and new password are passed to the user exit. If user exit 1 is not active, the daemon checks whether user name and password are correct for the system. If a new password is sent, the daemon changes the password by calling the Linux command `passwd`.
`[SSL]`	The key `ssl` is used to define whether the SSL protocol is to be used. One of the following values can be specified: "yes" or "no". See also *SSL Support*.
`[PAM]`	The key `pam` is used to define whether the PAM (Pluggable Authentication Modules) mechanism is to be used. One of the following values can be specified: "yes" or "no".

Section in Configuration File	Description
	PAM itself also has a configuration file or section (depends on the PAM implementation); the PAM configuration name must be `nwosrvd`.
`[HA]`	The key `ha` is used to define whether HA protocol is to be used. One of the following values can be specified: "yes" or "no".

nwosrvd.sh - Shell Script for Starting and Stopping the Natural Web I/O Interface Daemon

The shell script *nwosrvd.sh* is used to start and stop the Natural Web I/O Interface daemon. For further information, see ***Starting and Stopping the Natural Web I/O Interface Daemon***.

nwo.sh - Shell Script for Starting Natural

In order to start a Natural session, the Natural Web I/O Interface service executes a shell script. The shell script prepares the environment for the Natural session and eventually starts Natural. It must therefore contain all environment settings needed to run the Natural session.

The shell script receives certain parameters from the Natural Web I/O Interface client. The parameters can either be evaluated by the shell script itself or passed on to Natural. A client who wants to start a Natural session can specify the shell script to be used.

The shell script *nwo.sh* is called from the Natural Web I/O Interface daemon in order to start a Natural session. It has the following content:

```
#!/bin/sh

echo "Number of arguments $#" > nwo.log

IPAddress=""
ClientId=""
CodePage=""
CustomParameters=""
NaturalParameters=""

if [ "$1" != "null" ]
then
  IPAddress="$1"
fi

if [ "$2" != "null" ]
then
  ClientId="$2"
fi

if [ "$3" != "null" ]
then
  CodePage="$3"
```

```
fi

if [ "$4" != "null" ]
then
  CustomParameters="$4"
fi

if [ "$5" != "null" ]
then
  NaturalParameters="$5"
fi

#echo "IP Address="$IPAddress >> nwo.log
#echo "Client Id="$ClientId >> nwo.log
#echo "Code Page="$CodePage >> nwo.log
#echo "Custom Parameters="$CustomParameters >> nwo.log
#echo "Natural Parameters="$NaturalParameters >> nwo.log
#echo "NWO_BROWSER_IO="$NWO_BROWSER_IO >> nwo.log

$NAT_HOME/bin/natural $NaturalParameters etid=$$ > /dev/null 2>&1
```

You have to create such a shell script for each Natural application. It can have any name and it must be located in a directory which is defined in the environment variable `PATH`.

The name of the shell script is taken from the configuration file for the session. It is taken from the configuration file section that is defined for the session that the user has selected in the logon page. For further information, see *Configuring the Client*.

Arguments

The shell script will receive the following arguments:

Order	Argument	Description
1	`IPAddress`	The client IP address from where the session is opened. **Note:** If there is a proxy, this will not be the IP address of the client workstation. Instead, it will be the IP address of the proxy.
2	`ClientId`	The user name from the logon page is passed as the client ID.
3	`CodePage`	The encoding that is defined in the configuration file for the session. This value can be used to set the Natural system variable `*CODEPAGE`.
4	`CustomParameters`	From the logon page, it is possible to pass any values to the script in order to execute any desired action. Example: you pass a small text to the script which describes an error. When the script receives this error text, it sends it as an e-mail to the administrator.
5	`NaturalParameters`	These can be any Natural parameters. The parameters are either defined in the configuration file for the session, or they are entered in the logon page.

Order	Argument	Description
		The following is an example of the corresponding entry in the configuration file: `<natural_parameter>parm=nwoparm\ stack=(logon\ mylib;start-program;fin)<natural_parameter>` The language that is selected in the logon page is added as the first element to the Natural parameters in the form "ulang=x".

Arguments 1 to 4 can be used to audit the client, to allow to run an application from a specific PC (identifying the IP address), to build statistics, to do special actions, etc.

Environment Variables

In the shell script, several environment variables can be set for the Natural session that is started by the daemon:

NWO_ENABLE_ACK=["YES"|"NO"]

This environment variable is used for asynchronous screens (`SET CONTROL N`).

YES	When asynchronous screens are sent to the client, Natural will wait to receive an ACK package before the next screen can be sent.
NO	No waiting between asynchronous screens. Default value.

NWO_PF_MSG_LINES_NATIVE_FORMAT=["YES"|"NO"]

This environment variable defines how the PF keys and the message line are to be shown.

YES	The PF key prompting lines and the message line are shown as output text, as in the native Linux environment.
NO	The PF keys are rendered as buttons and the message line is rendered as a special message line element. Default value.

NWO_TIMEOUT=[*number-of-seconds*]

The maximum time, in seconds, that Natural waits to receive any input from the client before it closes the session. If the number of seconds is "0", Natural waits infinitely (no timeout). The default value is "0".

Error NAT5466 is returned at timeout. In Natural, the application can handle this error and decide how to continue or terminate.

Logging Information

The logging information system reports errors, warnings and/or session information, depending on the option that has been defined with the following **Natural Web I/O Interface daemon command**:

```
nwosrvd -config -logging=option
```

`option` can be one of the following:

Option	Description
`E`	Error. When this option is specified, the Natural Web I/O Interface daemon reports only errors. In the case of an error, the daemon usually exits immediately.
`W`	Warning. When this option is specified, the Natural Web I/O Interface daemon reports errors and warnings for uncritical errors. In the case of a warning, the daemon continues to run.
`I`	Information. When this option is specified, the Natural Web I/O Interface daemon reports errors, warnings and information. The information messages allow to check the session parameters, IP address, etc.

Help information, for example, on how to run, configure and install the Natural Web I/O Interface daemon is always provided. The messages which inform you when the daemon has been started or stopped are also part of the help information.

To find out which logging option is currently active, enter the following Natural Web I/O Interface daemon command:

```
nwosrvd -show
```

The logging messages are shown directly for the standard output. The format of the messages is as in the following example:

```
%NWOSRVD-E: 18.01.2008 14:55:20 NWO_SRVDCONF is not established.
```

The following information is provided:

- `%NWOSRVD` is the internal name of the Natural Web I/O Interface daemon.
- The message type is shown directly after `%NWOSRVD`. It can be one of the following: `-E` (error), `-W` (warning), `-I` (information), or `-H` (help).
- Date and time when the message was reported.
- Any text or message which pertains to the error, warning, information or help.

If you want to save these messages, you have to redirect the standard output to a file.

Example for csh:

```
nwosrvd 5454 >& nwosrvd_5454.log
```

Example for sh, ksh and bsh:

```
nwosrvd 5454 >& nwosrvd_5454.log 2>&1
```

SSL Support

SSL is used for a secure connection between the Natural Web I/O Interface server and the Natural Web I/O Interface client or Natural for Ajax. Server authentication cannot be switched off. A certificate and a private key is always required on the server.

To establish an SSL connection, you have to proceed as described in the following topics:

- Creating an SSL Certificate and a Private Key
- Configuring the Daemon
- Configuring the Client

Creating an SSL Certificate and a Private Key

To create and use an SSL certificate and a private key on the server, proceed as described below.

1. Adapt the example configuration file *openssl.cnf* to your needs.

   Note: *openssl.cnf* is delivered in `<install-dir>`*/common/security/openssl* and *openssl* is delivered in `<install-dir>`*/common/security/openssl/bin*.

2. Set the environment variable so that it points to the file `openssl.cnf`:

```
set OPENSSL_CONF=<install-dir>/common/security/openssl/openssl.cnf
export OPENSSL_CONF;
```

3. Generate a certificate signing request:

   ```
   openssl req -new > server.cert.csr
   ```

4. Generate a private RSA key:

   ```
   openssl rsa -in privkey.pem -out server.cert.key
   ```

5. Generate a self-signed certificate:

   ```
   openssl x509 -in server.cert.csr -out server.cert.crt -req -signkey ↩
   server.cert.key -days 365
   ```

   It is important that the name of the generated certificate is *server.cert.crt* and that the name of the generated private key is *server.cert.key*.

   Note: The certificate can be self-signed or it can be signed by a CA (Certificate Authority) such as VeriSign.

6. Put the generated files into the same directory as the scripts which start the Natural Web I/O Interface server.

Configuring the Daemon

After you have created an SSL certificate and a private key as described above, proceed as follows:

1. Change the configuration of the Natural Web I/O Interface daemon using the following command:

   ```
   nwosrvd -config -ssl=yes
   ```

2. Restart the Natural Web I/O Interface daemon.

See also *Configuring the Natural Web I/O Interface Daemon on Linux.*

Configuring the Client

After you have configured the daemon as described above, you have to import the generated *server.cert.crt* file to a truststore on the client. For information on how to do this for the Natural Web I/O Interface client, see *Configuring SSL*. If you are using Natural for Ajax as the client, see *Configuring SSL* in the Natural for Ajax documentation.

Working with the Linux Components of the Natural Web I/O Interface

The Linux components of the Natural Web I/O Interface are used to start the Natural applications linked with the Natural Web I/O Interface library.

The following topics are covered below:

- Starting and Stopping the Natural Web I/O Interface Daemon
- Starting a Natural Application

Starting and Stopping the Natural Web I/O Interface Daemon

The Natural Web I/O Interface daemons are responsible for accepting new sessions.

Since the daemon checks the user name and password, the following permissions must be set as follows (for setting the permissions, you must be super-user):

```
chmod 6755 nwosrvd.sh
```

```
chown root nwosrvd.sh
```

The Natural installation attempts to set permissions and owner. However, you have to verify this before you start the Natural Web I/O Interface daemon.

The daemon can be started and stopped using the following command:

```
cd $NAT_HOME/nwo/$NWONODE
nwosrvd.sh portnumber [start|stop]
```

Alternatively:

```
cd $NAT_HOME/INSTALL
nwosrvd.bsh  [start|stop]
```

Note: The daemon must be started on a port which is not yet used.

The shell script you have created must be in the same directory as the *nwosrvd.sh* script. It will be used by the Natural Web I/O Interface (configuration file for the session; see ***Configuring the Client***). The following is an example of the corresponding entry in the configuration file:

```
<natural_program>your-shell-script.sh</natural_program>
```

Starting a Natural Application

Almost any Natural application can be used with the Natural Web I/O Interface. See also *Differences between the Natural Web I/O Interface Client and Terminal Emulation*.

To start a new Natural application with the Natural Web I/O Interface, proceed as follows:

1. Create a new parameter file from `NWOPARM` using the Configuration Utility.
2. In this new parameter file, modify the `STACK` command as follows:

   ```
   logon library; startprogram; fin
   ```

   Note: Only "real" Natural applications can be used. The Natural **Main Menu** cannot be used as a Natural application.

Add the new service as follows:

1. Look for a port number which is not yet used.
2. Create a new shell script (similar to *nwo.sh*) for starting the Natural application:

   ```
   cd $NAT_HOME/nwo/$NWONODE
   copy nwo.sh your-shell-script.sh
   vi your-shell-script.sh
   ```

   You have to decide which (last) line you will use in the script. Use one of the following:

   ```
   $NAT_HOME/bin/natural parm=parameter-file etid=$$ >output-file 2>&1
   ```

   ```
   $NAT_HOME/bin/natural $5 etid=$$ >output-file 2>&1
   ```

   When using the line with `parm=parameter-file`, the above step in which you modify the `STACK` command is mandatory.

   When using `$5`, the Natural parameter (`parameter-file` and `STACK` command) is taken from the configuration file for the session (see *Configuring the Client*). The following is an example of the corresponding entry in the configuration file:

   ```
   <natural_parameter>parm=myparm stack=(logon mylib;menu;fin)<natural_parameter>
   ```

3. If you want to define special settings for the Natural session, you can set the environment variables in your shell script. See **above**.
4. Set the permissions for the shell script which starts the service as follows:

   ```
   chmod 775 script-name
   ```

The service is now available for use with a PC.

III 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:

4 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.

5 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***.

6 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>
```

IV 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:

7 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:

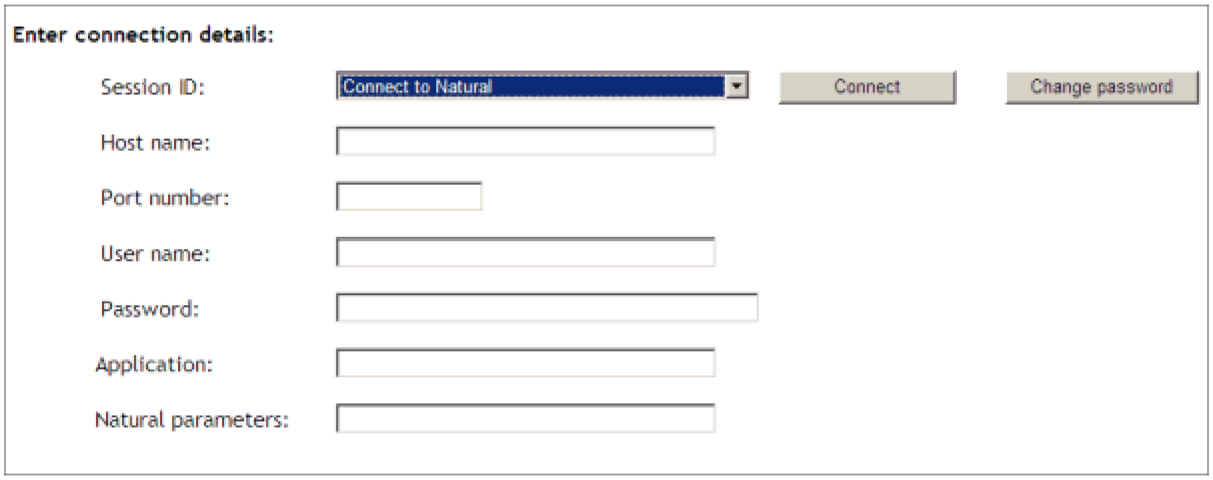
```
http://<host>:<port>/natuniweb/natural.jsp
```

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

Examples of Logon Pages

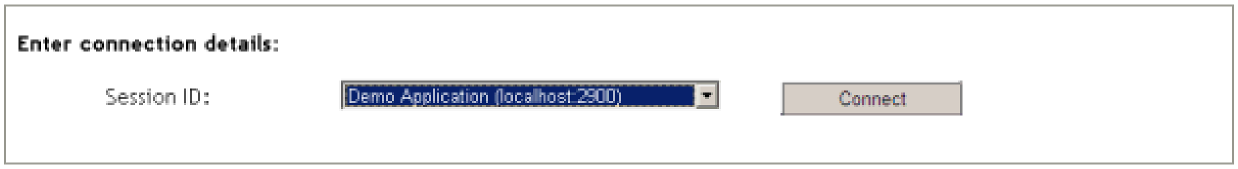
For each session definition that has been configured in the configuration file, an entry appears on the logon page. If the user selects the corresponding entry, only those parameters that were not pre-configured in the configuration file need to be specified in the logon page in order to start the application. 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:


Enter connection details:

Session ID:	Connect to Natural
Host name:	
Port number:	
User name:	
Password:	
Application:	
Natural parameters:	

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.

8 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

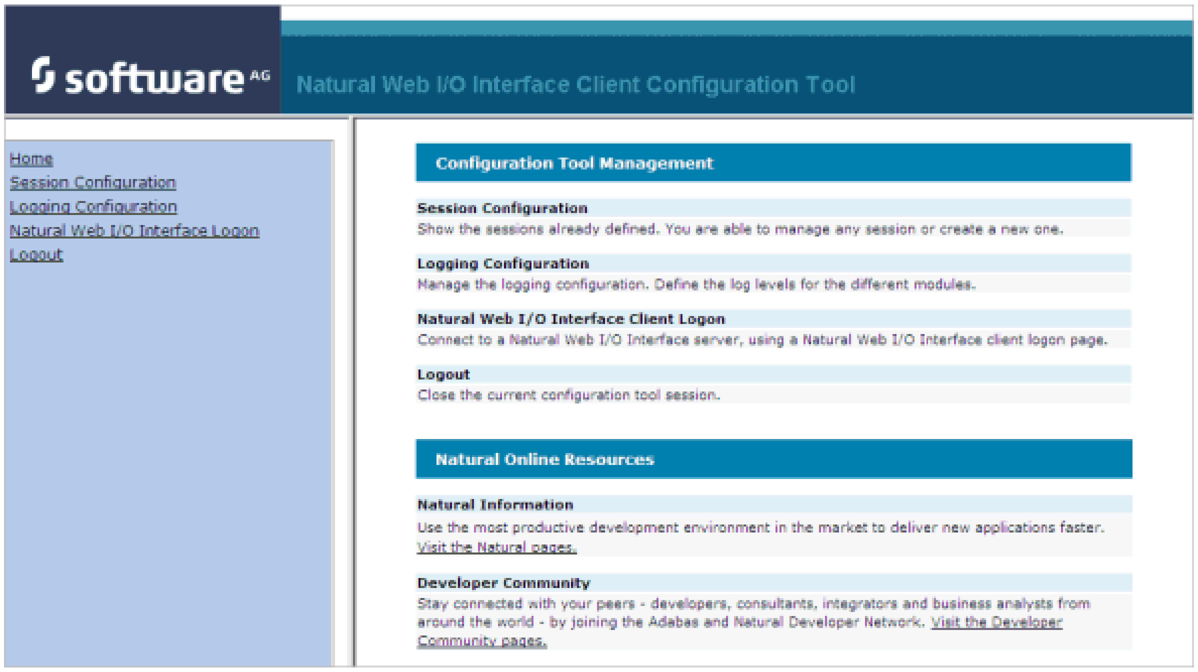
- Enter the following URL in your browser:

```
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.


software AG
Natural Web I/O Interface Client Configuration Tool
Home
Session Configuration
Logging Configuration
Natural Web I/O Interface Logon
Logout

Configuration Tool Management

Session Configuration
Show the sessions already defined. You are able to manage any session or create a new one.

Logging Configuration
Manage the logging configuration. Define the log levels for the different modules.

Natural Web I/O Interface Client Logon
Connect to a Natural Web I/O Interface server, using a Natural Web I/O Interface client logon page.

Logout
Close the current configuration tool session.

Natural Online Resources

Natural Information
Use the most productive development environment in the market to deliver new applications faster. Visit the Natural pages.

Developer Community
Stay connected with your peers - developers, consultants, integrators and business analysts from around the world - by joining the Adabas and Natural Developer Network. Visit the Developer Community pages.

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:

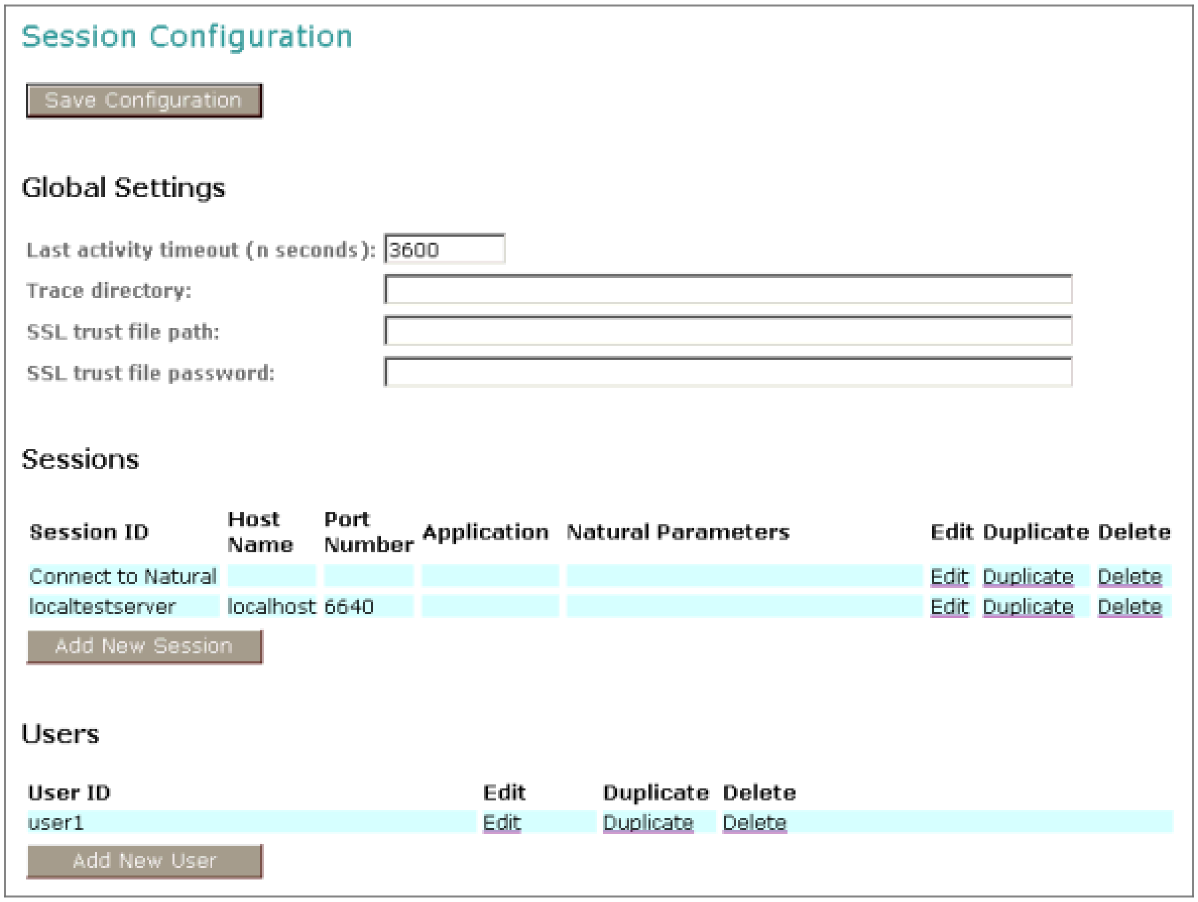
Invoking the Session Configuration Page

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:



Session Configuration

Global Settings

Last activity timeout (n seconds):	3600
Trace directory:	
SSL trust file path:	
SSL trust file password:	

Sessions

Session ID	Host Name	Port Number	Application	Natural Parameters	Edit	Duplicate	Delete
Connect to Natural					Edit	Duplicate	Delete
localtestserver	localhost	6640			Edit	Duplicate	Delete

Users

User ID	Edit	Duplicate	Delete
user1	Edit	Duplicate	Delete

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.

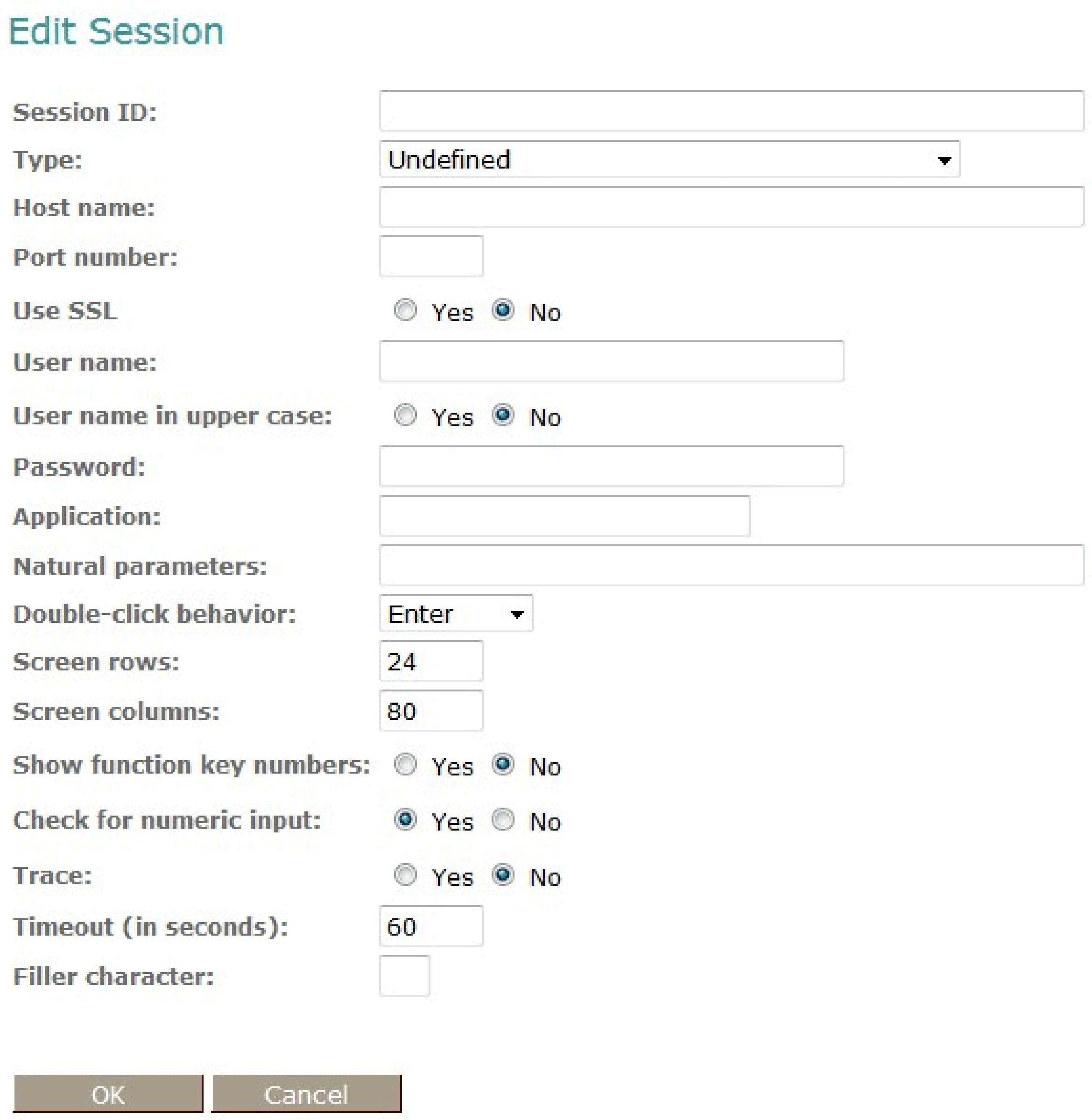
4 Choose the **Save Configuration** button to write the modifications to the configuration file.

Overview of Session Options

The **Edit Session** page appears when you

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

Example:



Edit Session

Session ID:	
Type:	Undefined
Host name:	
Port number:	
Use SSL	Yes [✓ No]
User name:	
User name in upper case:	Yes [✓ No]
Password:	
Application:	
Natural parameters:	
Double-click behavior:	Enter
Screen rows:	24
Screen columns:	80
Show function key numbers:	Yes [✓ No]
Check for numeric input:	[✓ Yes] No
Trace:	Yes [✓ No]
Timeout (in seconds):	60
Filler character:	

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.

9 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.

`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. 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.

10 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. Both, "own" and standard runtime class loaders 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.

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

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.

11 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.

12 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".

- 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.

13 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=my-username

14 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*** .

15 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* .

16 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*

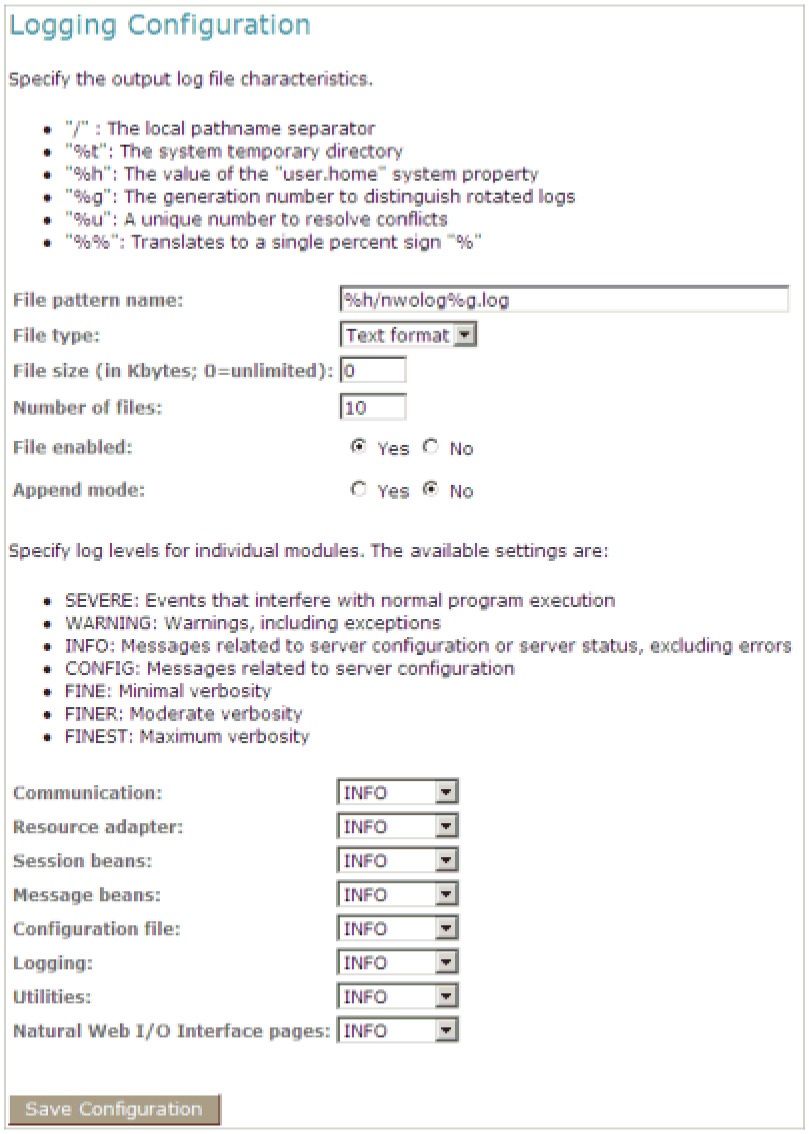
Invoking the Logging Configuration Page

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:



Logging Configuration

Specify the output log file characteristics.

- "/" : The local pathname separator
- "%t": The system temporary directory
- "%h": The value of the "user.home" system property
- "%g": The generation number to distinguish rotated logs
- "%u": A unique number to resolve conflicts
- "%%": Translates to a single percent sign "%"

File pattern name:	%h/nwolog%g.log
File type:	Text format
File size (in Kbytes; 0=unlimited):	0
Number of files:	10
File enabled:	[✓ Yes] No
Append mode:	Yes [✓ No]

Specify log levels for individual modules. The available settings are:

- SEVERE: Events that interfere with normal program execution
- WARNING: Warnings, including exceptions
- INFO: Messages related to server configuration or server status, excluding errors
- CONFIG: Messages related to server configuration
- FINE: Minimal verbosity
- FINER: Moderate verbosity
- FINEST: Maximum verbosity

Communication:	INFO
Resource adapter:	INFO
Session beans:	INFO
Message beans:	INFO
Configuration file:	INFO
Logging:	INFO
Utilities:	INFO
Natural Web I/O Interface pages:	INFO

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.