Natural for Ajax

Client Configuration

Version 9.1.1

October 2018

This document applies to Natural for Ajax Version 9.1.1 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: NJX-NNJXCLIENT-911-20181002

Table of Contents

Preface

This documentation explains how to configure Natural for Ajax so that it can be used in a Natural runtime environment. The following topics are covered:

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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

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.

2 About the Natural Web I/O Interface

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

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 which is part of the Natural documentation for the different platforms.
- 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. For further information on the Natural Web I/O Interface server, see the Natural documentation for the different platforms.

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.

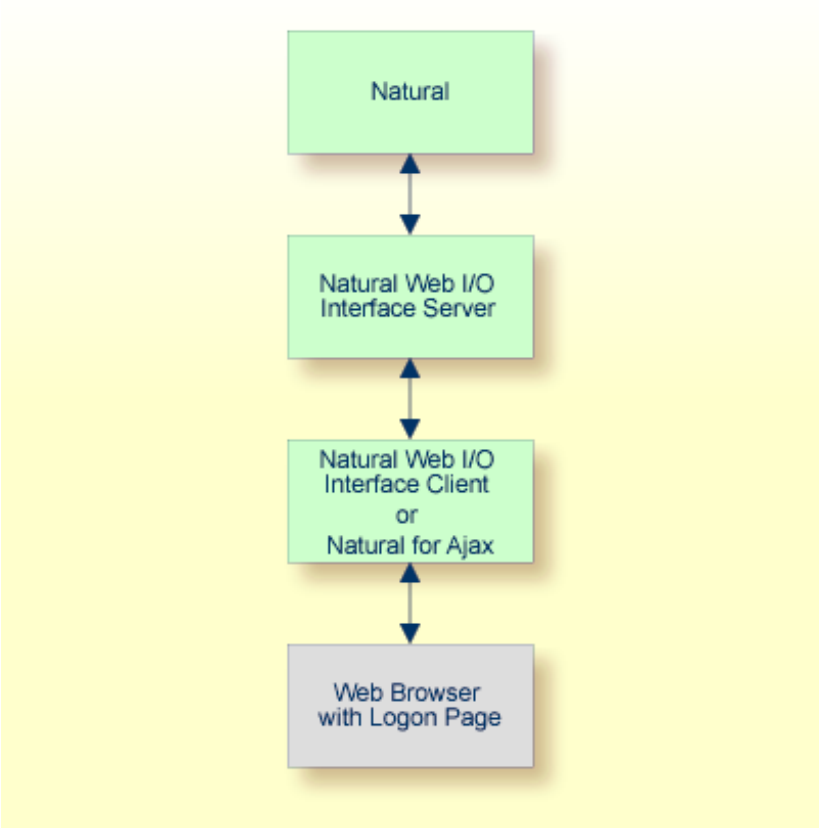
Two types of client are supported:

- 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. For further information on this type of client, see the Natural documentation for the different platforms.
- Natural for Ajax for displaying rich internet applications in the web browser.

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

Executing a Natural Application in a Web Browser

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



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

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

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

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 UNIX and Natural for OpenVMS, 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.

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 UNIX, OpenVMS, 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 UNIX 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
```

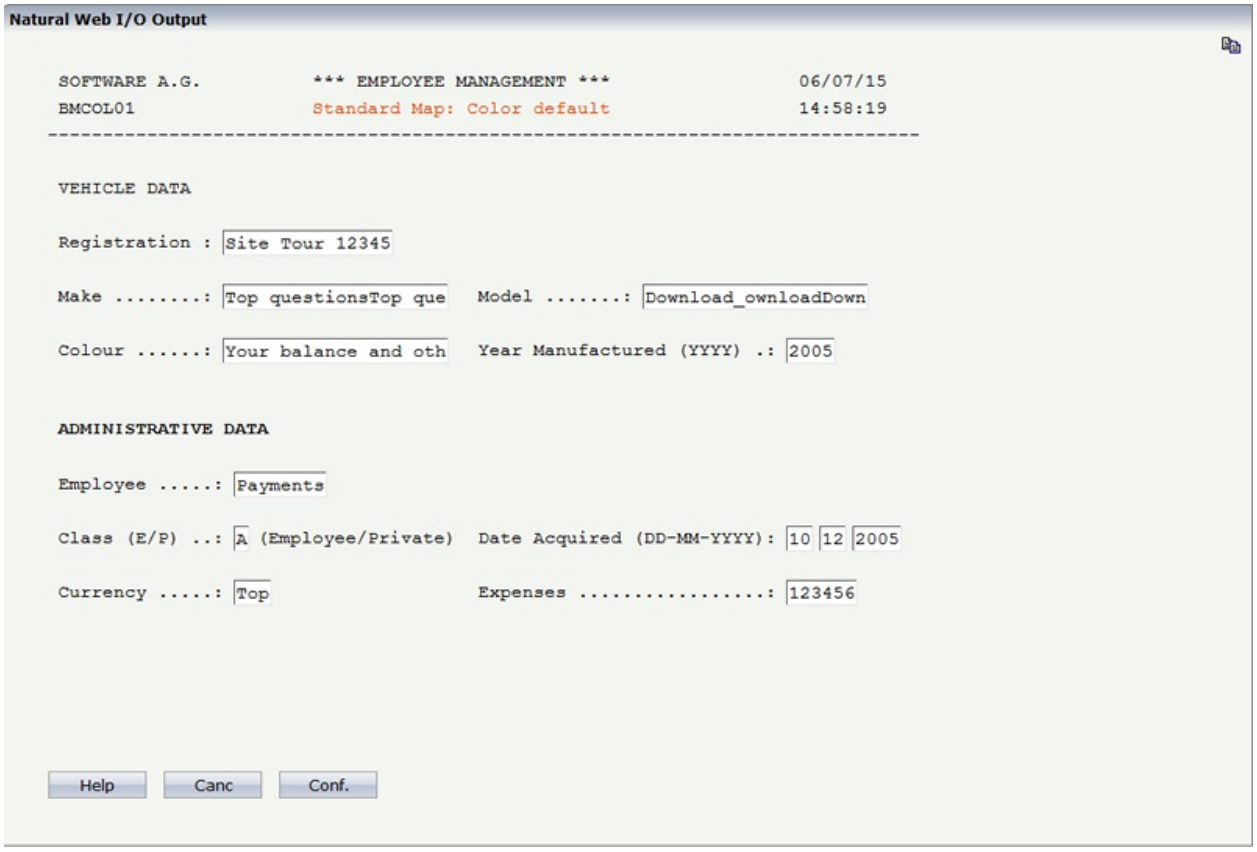
- **Terminal commands**
  Terminal commands are not supported. They do not work when entered in the Natural for Ajax client.
- **Natural system variable** `*INIT-ID`
  When using the Natural for Ajax client with Natural applications on UNIX, OpenVMS, mainframe or Windows hosts, the Natural system variable `*INIT-ID` will not be filled with a value for the terminal type. On UNIX, OpenVMS 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 UNIX, OpenVMS 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.

3 Natural Web I/O Screens

Whenever you display character-based Natural application screens in Natural for Ajax, they are rendered as web I/O pages:



In web I/O mode, certain basic functions behave differently for technical reasons.

Copying Screen Content in Web I/O mode

To copy selected lines or the complete screen to the clipboard in the web I/O mode, you have to switch from the standard web I/O screen to a copy mode output screen. This can be done by clicking on the following icon on the top right of the web I/O output screen:

Alternatively, you can use the shortcut **CTRL+Y** to activate the copy mode.

When the web I/O output is in the copy mode, it is possible to select and copy the output text and use the standard **CTRL+C** combination or the browser´s **Copy** menu item to copy the selected text to the clipboard. When the web I/O output is in copy mode, the icon on the top right of the screen switches to the following:

After copying the text to the clipboard, you may again click on the copy icon on the top right or use the shortcut **CTRL+Y** to return to the regular web I/O mode.

Switching to Another Style Sheet During the Session

If enabled in the configuration file for the session, a user can switch to another style sheet during a running session. In this case, the user can open the **Style Sheet** control in the output window.

To switch to another style sheet, the user has to select it from the drop-down list box and then choose the **Apply** button.

4 About the Logon Page

Starting a Natural Application from the Logon Page

When you start Natural for Ajax in the browser, a logon page appears. The entries in this logon page depend on the settings in your configuration file (see *Natural Client Session Configuration*).

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

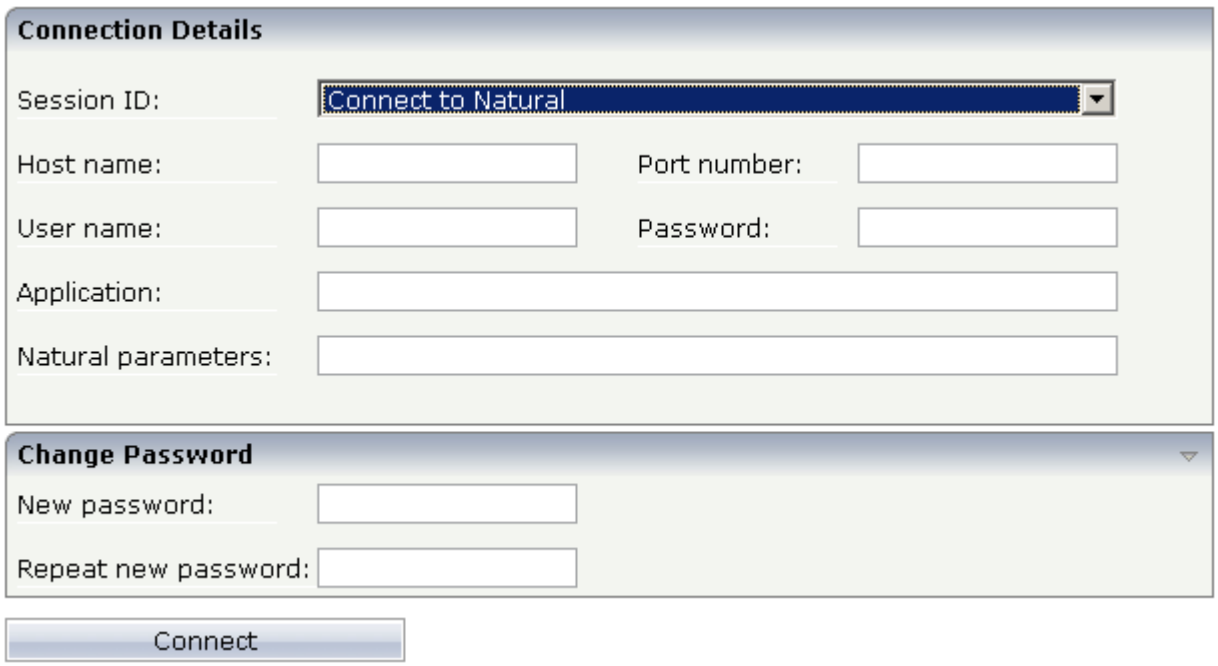
```
http://<host>:<port>/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

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

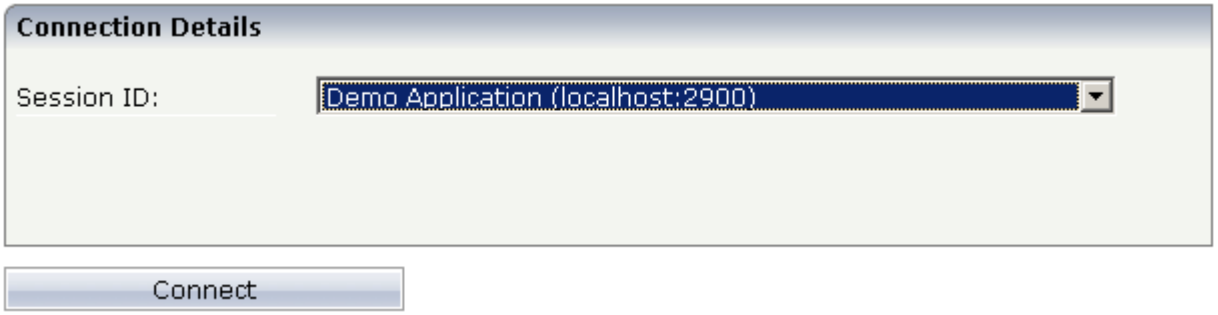
Examples of Logon Pages

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 preconfigured in the configuration file need to be specified in the logon page in order to start the application. Usually, you will preconfigure all connection parameters except user name and password.

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



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



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

Dynamically Changing the CICS Transaction Name when Starting a Session

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

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

```
<TA_NAME=name>
```

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

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

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

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

Note: The above definition for the CICS transaction name can also be specified in the **configuration tool**, in the same place where you also specify the Natural parameters, and together with the **URL parameter** `xciParameters.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, UNIX and OpenVMS, 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 UNIX, Natural for OpenVMS 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 Click the title **Change Password** to show the content of this input area.

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.

Browser Restrictions

The browser's “Back” and “Forward” buttons do not work with Natural for Ajax 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.

5 Overview of Configuration Files

Natural for Ajax applications have the following three central configuration settings:

- Natural client-specific settings are defined in the file *sessions.xml*, see ***Natural Client Session Configuration***.
- Ajax-specific settings for customizing the rendering of pages and controls are defined in the file *cisconfig.xml*, see ***Ajax Configuration***.
- General web application settings of a Natural for Ajax web application are defined in the file *web.xml*.

6 Natural Client Session Configuration

General Information

The configuration file is an XML file which is required to define the sessions that can be invoked from the logon page.

To edit the configuration file, you 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 *Natural Client Configuration Tool* for further information.

Name and Location of the Configuration File

The name of the configuration file is *sessions.xml*. It can be found in the *WEB-INF* directory. The path to this directory depends on the application server that you are using.

- **Wildfly Application Server**
  `<application-server-install-dir>`*/webapps/cisnatural.war/WEB-INF*
- **Apache Tomcat**
  `<tomacat-install-dir>`*/webapps/cisnatural/WEB-INF*

7 Natural Client Configuration Tool

Invoking the Configuration Tool

Natural for Ajax 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 Natural for Ajax.

To invoke the configuration tool

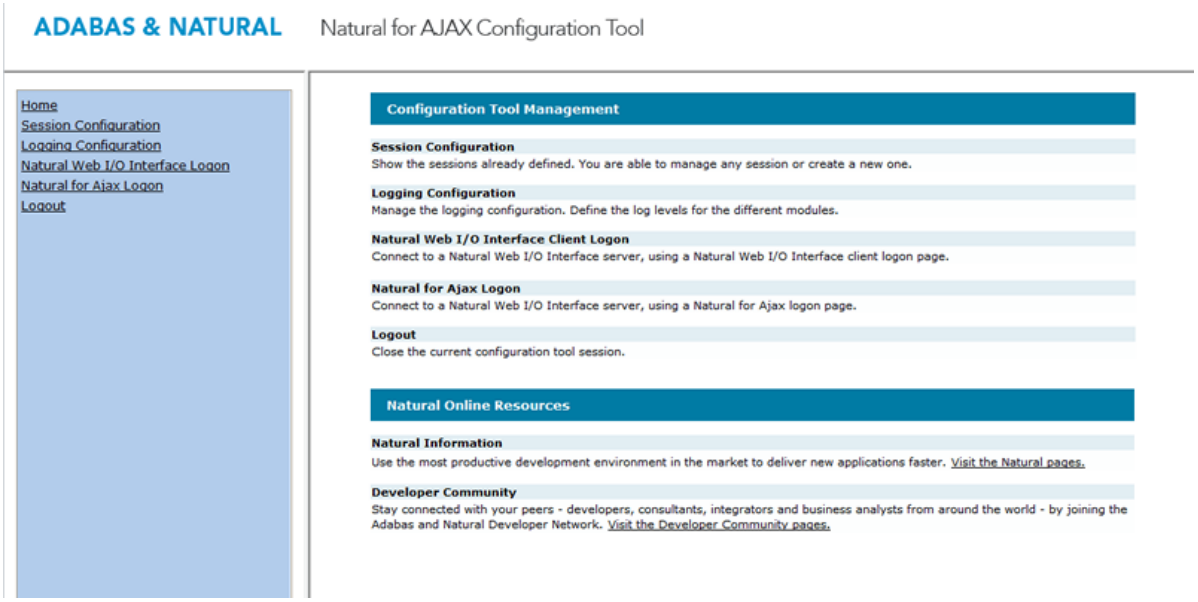
- Enter the following URL in your browser:

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

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

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

  The configuration tool appears.



  The configuration tool has two frames.

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

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

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

Session Configuration

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

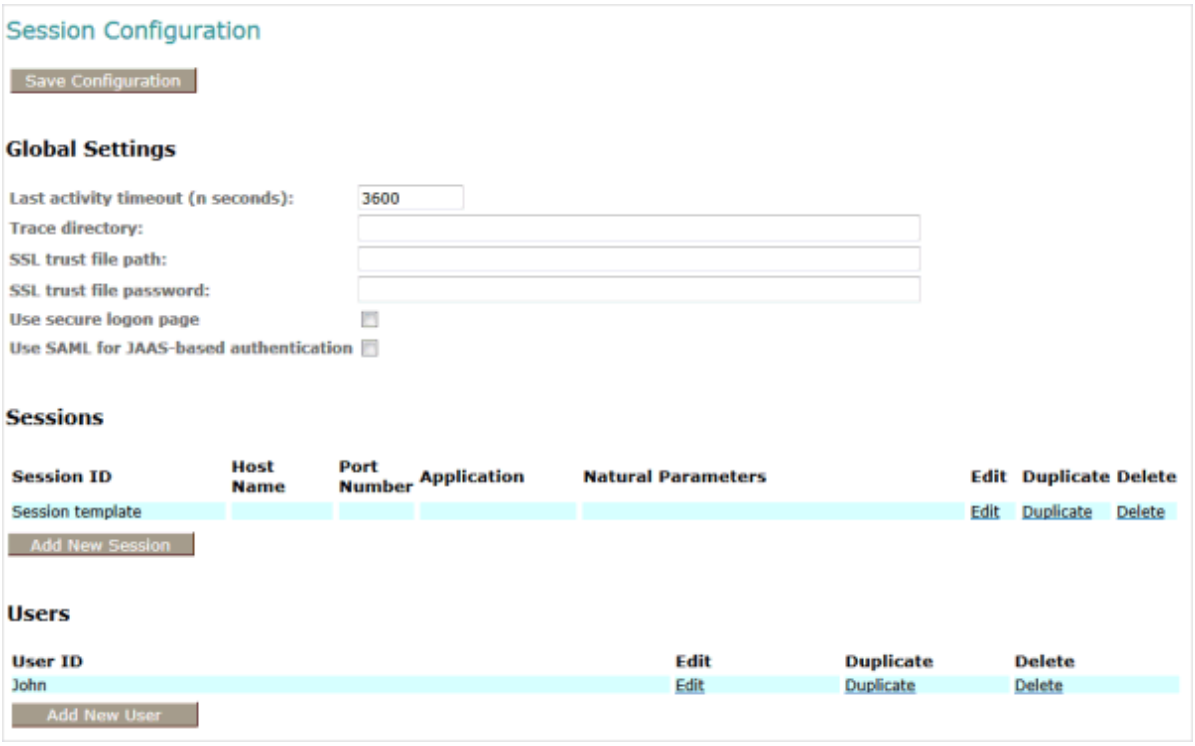
Invoking the Session Configuration Page

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

To invoke the Session Configuration page

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

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



Global Settings

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

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

Option	Description

	`set JAVA_OPTS=%JAVA_OPTS% -Djava.io.tmpdir=C:\temp`
	
	■ Example for UNIX (*run.sh*):
	
	`set JAVA_OPTS="$JAVA_OPTS -Djava.io.tmpdir=/tmp`
	
	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.
Use secure logon page	If selected, users are authenticated before they access the Natural for Ajax logon page. See also *Configuring Application-Managed Authentication*. If **Use SAML for JAAS-based authentication** is also selected, the secure logon page is only shown in the case of an error in the SAML artifact validation.
Use SAML for JAAS-based authentication	If selected, Natural for Ajax expects the use of the `SAMLArtifactLoginModule` which does not prompt the end user for a user name and password. If you want to use other login modules which expect the user to enter a valid user name and password (such as the `SSXLoginModule`), make sure that this check box is not selected. See also *Using Software AG Security Infrastructure*.

Adding a New Session

You can add a new session to the configuration file.

To add a new session

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

The **Edit Session** page appears.

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

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

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

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

Editing a Session

You can edit any existing session in the configuration file.

To edit a session

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

The **Edit Session** page appears.

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

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

The modifications are not yet available in the configuration file.

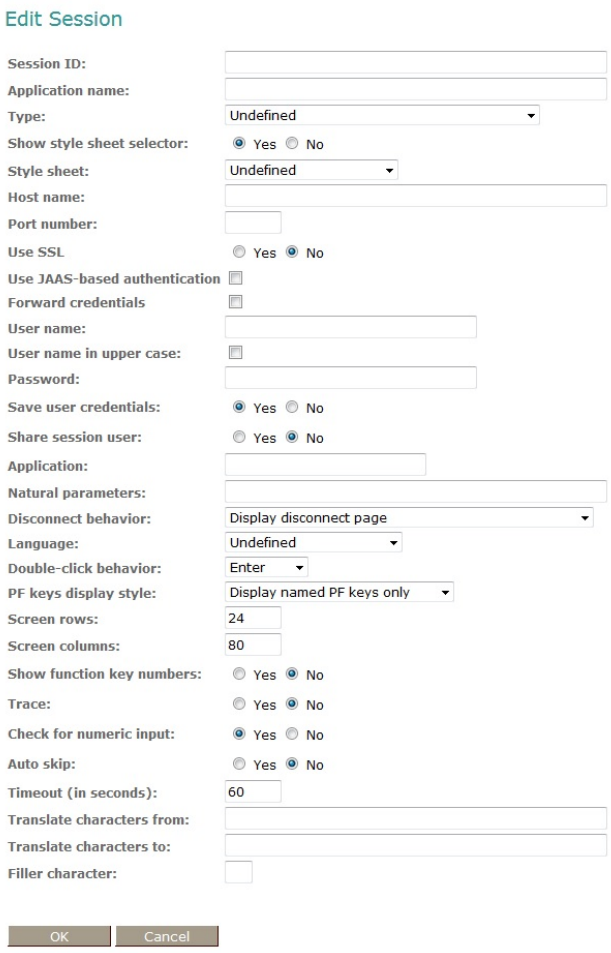
4 Choose the **Save Configuration** button to write the modifications to the configuration file.

Overview of Session Options

The **Edit Session** page appears when you

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

Example:



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

Option	Description
Session ID	Mandatory. A session name of your choice. On the logon page, the session name is provided in a drop-down list box.
Application name	Optional. An application name of your choice. This name is used to provide more individual disconnect messages. The name entered here will replace the term "Application" in the standard disconnect message of Natural for Ajax applications. For example, if the field **Application name** contains the word "Calculator" the disconnect message "Application finished successfully" would instead be "Calculator finished successfully". Similar custom messages will be displayed if the application ends with an error.

Option	Description
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, UNIX or OpenVMS 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, UNIX or OpenVMS** 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*").
Show style sheet selector	With Natural for Ajax, the users can switch to another style sheet during a running session. If set to **No**, the users are no longer able to select another style sheet.
Style sheet	The name of the style sheet which determines the colors, fonts and PF key button style of the current session. See ***Using Style Sheets***. When this element is specified, a fixed style sheet is used. In this case, the corresponding field does not appear on the logon page and the user is thus not able to select a different style sheet.
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 Natural for Ajax 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, UNIX or OpenVMS**. The other way around, if you want to use SSL with Natural for Windows, UNIX or OpenVMS, you must not select one of the mainframe types; the type may also be **Undefined** in this case.
Use JAAS-based authentication	If selected, application-managed authentication is used. This setting implies that **Forward credentials** is also selected. See also ***Configuring Application Managed Authentication***. If **Use SAML for JAAS-based authentication** is also selected in the global settings, the user name and password are not visible in the Natural for Ajax logon page. See also ***Using Software AG Security Infrastructure***.
Forward credentials	If selected, the security credentials from the application server are forwarded to the Natural Web I/O Interface server, thus sparing the user a second logon. See also ***Forwarding the User Credentials to Natural***.
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.

Option	Description
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, UNIX and OpenVMS, 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. When a password is specified, the corresponding field does not appear on the logon page. The configuration tool saves the password in encrypted form.
Save user credentials	Applies only to applications that are designed as Application Designer workplaces. If set to **Yes** (default), the default behavior of the option **Share session user** applies. If set to **No**, the user credentials (user ID and password) are not saved in the Application Designer session and are therefore not available for an Application Designer subsession. An example for a workplace application is available under the following URL: *http://<host>:<port>/cisnatural/servlet/StartCISPage?PAGEURL=/njxdemos/wpworkplace.html* where *<host>* and *<port>* are the host name and port number of your application server.
Share session user	Applies only to applications that are designed as Application Designer workplaces. If set to **No** (default), the user credentials of the main Application Designer session are automatically used in an Application Designer subsession if the server and port of the subsession is the same as in the main session. If the server and port are not the same, the user has to specify the user ID and password in a logon dialog. If set to **Yes**, the user credentials of the Application Designer main session are always used for all Application Designer subsessions on all involved servers - even if the server and port are different.
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 UNIX** The name of the UNIX shell script for starting the Natural application (a file similar to *nwo.sh*). ■ **Natural for OpenVMS** The name of the Natural image file (for example, *natural<version>* or *natural<version>.exe*).

Option	Description
	■ **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.

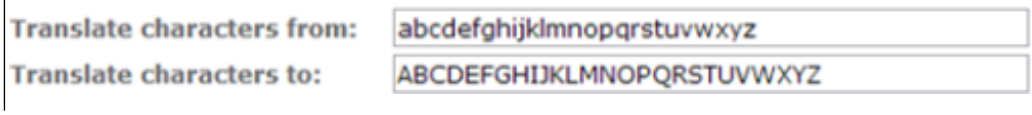
Natural parameters	Optional. Parameters for starting the Natural application. This can be stack parameters, a parameter file/module or other Natural-specific information.
	
	■ **Natural for Mainframes**
	Used to pass dynamic Natural profile parameters to the session, for example:
	
	`SYSPARM=(MYPARMS) 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 UNIX and Natural for Windows**
	Used when the above shell script (UNIX) or command file (Windows) uses the parameter `$5` after "natural", for example:
	
	`PARM=MYPARM STACK=(LOGON MYLIB;MENU)`
	
	■ **Natural for OpenVMS**
	Used for starting a Natural application, for example:
	
	`BP=BP`*`node-name`* `NLDCHK WEBIO=ON "STACK=(LOGON SYSEXT;MENU)"`

Disconnect behavior	Defines whether the default disconnect page is to be shown when a Natural program ends, or whether the current browser tab is to be closed. You can select the required setting from the drop-down list box. ■ **Display disconnect page** Default. The disconnect page is always displayed. ■ **Always close browser** The current browser tab is closed. When the last tab in the browser is closed, the entire browser is closed. ■ **Close browser, but display disconnect page on error** The current browser tab is closed. When the last tab in the browser is closed, the entire browser is closed. However, in the case of an unexpected error, the disconnect page is shown. Internet Explorer allows closing the browser with JavaScript by default. For Firefox, however, this has to activated as follows: 1. In the Firefox address bar, type the following:

Option	Description

	`about:config`
	
	A warning message appears, saying that changing the advanced settings can be harmful and that you should only continue if you are sure of what you are doing.
	2. Choose the **I'll be careful, I promise!** button.
	3. In the **Filter** text box, type the following:
	
	`dom.allow_scripts_to_close_windows`
	
	4. Set this value to "true" by double-clicking on the entry (default: "false").

Language	You can select the required language from the drop-down list box. It is also possible to select **Set in workplace** from the the drop-down list box. When selected, the language that is currently used in the workplace will also be used for the next start of Natural. See also *Multi Language Management*. Default: English.
Double-click behavior	Applies only to Natural maps, not to rich GUI pages. 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.
PF keys display style	Applies only to Natural maps, not to rich GUI pages. By default, only the named function keys are shown as buttons. It is also possible to show buttons for all function keys, including those which do not have names. You can specify whether to display buttons for 12, 24, 36 or 48 function keys. Each line always contains 12 function key buttons. The first line also contains a button for the ENTER key. Each function key button is always displayed at the same position. You can select the required setting from the drop-down list box.
Screen rows	Applies only to Natural maps, not to rich GUI pages. 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	Applies only to Natural maps, not to rich GUI pages. 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	Applies only to Natural maps, not to rich GUI pages. 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.
Auto skip	Applies only to Natural maps, not to rich GUI pages. If set to **Yes**, the cursor is automatically placed in the next input field when the last possible character has been entered in the current input field. If set to **No** (default), the cursor remains in the current input field.
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.
Translate characters from / Translate characters to	Applies only to Natural maps, not to rich GUI pages. Optional. Used to translate characters that are sent from the host or entered by the user into different characters to be shown on the screen. You enter the characters to be translated in the **Translate characters from** text box and the characters to be used instead in the **Translate characters to** text box (in the same order as in the first text box). Both text boxes must contain the same amount of characters. The following example shows how to translate all lower-case characters to upper-case characters: 

Option	Description
	The following example shows how to translate the so-called “European numerals” to “Arabic-Indic numerals”: Translate characters from: 1234567890 Translate characters to: ١٢٣٤٥٦٧٨٩٠
Filler character	Applies only to Natural maps, not to rich GUI pages. 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/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html&xciParameters.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 links in the left frame:

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

Each of these links opens the corresponding 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.

8 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"
           useowncclassloader="true"
           browserpopuponerror="false"
           framebuffersize="3"
           ↩
onlinehelpmanager="com.softwareag.cis.onlinehelp.projectbased.FrameHelpOHManager"
           textencoding="UTF-8"
           enableadapterpreload="true">
</cisconfig>
```

`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.
`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.
`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.
`jsconsolelog`	Defaut: false. If set to true, the Ajax framework writes log information to the console of the browser. Only set this to true when asked by Software AG Support.
`loglevel`	Default: EWI. Defines the message types that are to be logged. Values: E (error) W (warning) I (information) D (debug) 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.

	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.
`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.

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.
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.
`startmonitoringthread`	Default: true. If set to "true", a monitoring thread is opened which by default wakes up every 5 seconds. You can customize this value by setting the parameter `monitoringthreadinterval`. The thread performs the following activities: 1. It initiates a garbage collection periodically (every two minutes). 2. It writes all log information into a log file (every *n* milliseconds. Where *n* represents the interval length defined in the `monitoringthreadinterval` parameter). 3. It calls the clean up of sessions which are timed out (every two minutes) 4. It checks for user interface component updates, which need to be deployed. See also *Deploying the User Interface Components*. What happens if the monitoring thread is not started? 1. No garbage collection will be triggered by Application Designer. This is then the task of the servlet container around. 2. The log is not automatically written to the file location specified in the *web.xml* file, but is written to the servlet container's logging.

	3. Timing out sessions is not done every two minutes but every thousand requests. 4. No user interface deployment will be done. **Caution:** Some servlet containers do not allow to let the web application start new threads (for example, the Sun reference implementations do so). For these containers, the parameter must be set to "false". Values: true/false.
`suppressfocusmanagement`	Default: false. If you set this parameter to "true", no focus management in the client will be done after a server round trip. This means: The focus will not be set to focus-requesting controls such as "EDIT" fields with "ERROR" status after a server round trip. Usually, you do not set this parameter. If you need to suppress focus management for specific server round trips, you usually do this from within your adapter code for these specific server round trips. See also the `focusmgtprop` in the NATPAGE control. Only set this parameter to "true" if your application needs to do it vice versa: Suppress focus mangement for nearly all server round trips and only explicitly activate focus management for some specific server round trips from within your adapter code. Values: true/false.
`takeoutfieldpopupicon`	Default: false. Set this parameter to "true" in case you are using right-aligned FIELD controls with value help. This will avoid overlapping of the right-aligned text and the corresponding drop-down icon. Values: true/false.
`testtoolidhtml4`	Default: false. If set to "true", the HTML attribute generated for the test tool IDs has the name "testtoolid". Otherwise, the name is "data-testtoolid". See also the information on standards mode and HTML5 in the Natural for Ajax documentation.
`textencoding`	Default: UTF-8. By default, Application Designer reads and writes text files in UTF-8 format. You can tell Application Designer to use a different format (for example, for writing XML layout definitions). But be very careful and very aware of what you are doing.
`urlbackbuttonpressed`	When the browser back button is pressed, in some cases the page is not synchronized with the server anymore and the session has to be

	closed. In these cases a default page is displayed. Instead of this default page you can define a URL to a custom page. Value: the URL of the page that is to be shown instead of the default page.
`urlsessiontimeout`	When Application Designer times out a session (see the `sessiontimeout` parameter) and the user tries to continue to work with the session, a page will be displayed inside the user's browser, indicating that a timeout happened with the user's session. By default, this page is an Application Designer page that you might not want to show to your application users. Value: the URL of the page that is to be shown instead of the default page.
`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.

9 Overview of Style Sheets

Both, web I/O pages and Natural for Ajax pages can be styled using CSS style sheets. The style sheets of web I/O pages have a completely different structure than the style sheets of Natural for Ajax pages. More details on the different style sheet types are given in ***Natural Web I/O Style Sheets*** and ***Ajax Pages Style Sheets***.

10 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).

Natural for Ajax is delivered with a number of predefined style sheets. The default style sheet is *natural.css*. If you would like to see the same colors in the output window as in the map editor, you can use the style sheet *natural_mapeditor.css* instead of the default style sheet.

The location of the style sheets depends on the application server that you are using.

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;
}
```

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.

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.

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.

XSLT Files

In addition to the CSS files described above, Natural for Ajax 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 following elements are affected:

- 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.

Note: The JavaScript file which is part of the above conversion is *natunicscript.js*. It is located in the *scripts* directory which can be found in the `<installdir>`/*cisnatural* directory.

The name of the default XSLT file is:

- *transuni.xsl* for all supported browsers.

The default XSLT file can be found in the following directory:

`<installdir>`/*cisnatural/web-inf*

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

Important: It is recommended that you do not change the above XSLT file. Software AG may change or correct the original XSLT transformations in new versions or service packs of the product.

You can copy your own XSLT file into the above directory. In this case, the file must have the following name:

- *usertransuni.xsl* for all supported browsers.

If this user file can be found when the server is started, it is read instead of the default XSLT file.

When you make changes to this file, you have to restart the server so that your changes become effective.

11 Ajax Pages Style Sheets

Use the Style Sheet Editor tool to develop your own application-specific style sheet.

You can find further hints on how to apply a specific styling to the different controls in the respective control description chapters, for example *Working with Containers* and *Working with Grids*.

12 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
`xciParameters.natsession`	**Session ID**
`xciParameters.natserver`	**Host name**
`xciParameters.natport`	**Port number**
`xciParameters.natuser`	**User name**
`xciParameters.natpassword`	**Password**
`xciParameters.natprog`	**Application**
`xciParameters.natparam`	**Natural parameters**
`xciParameters.natparamext`	**Natural parameters** The URL parameter `xciParameters.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 `xciParameters.natparamext` are added, separated by a space character. If you want to overrule the definition in the configuration file, use the URL parameter `xciParameters.natparam` instead.

URL Parameter	Corresponding Option in the Session Configuration
xciParameters.nattimeout	**Timeout (n seconds)**
xciParameters.savesessionuser	**Save user credentials**
xciParameters.sharesessionuser	**Share session user**

Important: All parameter values must be URL-encoded.

Example: In order to start the Natural program MENU-NJX from the library SYSEXNJX, while your application server is running on *myappserver:4711*, your Natural Web I/O Interface server is running on *mywebio:4712*, and the name of the Natural startup script is *nwo.sh*, you can use the following URL:

http://myappserver:4711/cisnatural/servlet/StartCISPage?PAGEURL=%2Fcisnatural%2FNatLogon.html&xciParameters.natserver=mywebio&xciParameters.natprog=nwo.sh&xciParameters.natport=4712&xciParameters.natparam=stack%3D%28logon+SYSEXNJX%3BMENU-NJX%3Bfin%29

Instead of adding the parameters to the URL, you can also use the HTTP method POST to set the parameters in an HTML form. Example:

```
<html>
<head>
<title>Start Natural for Ajax Session</title>
<script type="text/javascript">
function submitStart() {
document.forms["myform"].submit();
}
</script>
</head>
<body>
 <form id="myform" name="myform" action="servlet/StartCISPage" method="post">
  <input type="hidden" name="PAGEURL" value="/cisnatural/NatLogon.html" />
   Host: <input type="input" size="20" name="xciParameters.natserver" value="mywebio" ↩
/><br/>
   Port: <input type="input" size="10" name="xciParameters.natport" value="4712" ↩
/><br/>
   Natural program: <input type="input" size="20" name="xciParameters.natprog" ↩
value="nwo.sh" /><br/>
   Natural parameter: <input type="input" size="50" name="xciParameters.natparam" ↩
value="stack=(logon sysexnjx;menu)" />
 </form>
 <a href="#" onclick="submitStart()">Start Session</a>
 <div id="status">Click on Start Session</div>
</body>
</html>
```

When you write the above example code to a file named *startWithPost.html* which is located in the *cisnatural* main directory, you can start the form with the following URL:

http://myappserver:4711/cisnatural/startWithPost.html

13 Configuring Container-Managed Security

General Information

Natural for Ajax 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 Natural for Ajax. Other examples are the Application Designer development workplace contained in Natural for Ajax or 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 Natural for Ajax and to adapt them to your requirements. More information on the topics described in this section can be found, for instance, at ***http://www.wildfly.org/***.

Notes:

1. The recommended security method is application-managed authentication. For further information, see ***Configuring Application-Managed Authentication***.
2. Container-managed security is not supported for IBM WebSphere Application Server. Use application-managed authentication instead.

Name and Location of the Configuration File

Security is configured in the file *web.xml*. The path to this file depends on the application server.

- **Wildfly Application Server**
  `<application-server-install-dir>`*/server/default/deploy/njx*`<nn>`*.ear/cisnatural.war/WEB-INF*
- **Apache Tomcat**
  `<tomcat-install-dir>`*/webapps/cisnatural/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>
```

Choosing the Login Module (WildFly)

On WildFly, Natural for Ajax is installed as a web application (WAR file). See *Installing Natural for Ajax on WildFly*.

The configuration of WildFly as a so-called standalone server is described here.

All configuration (especially the security configuration) is centralized in the file *<application-server-install-dir>/standalone/configuration/standalone.xml*.

In order to create a sample JAAS-based security configuration, proceed as follows:

1. Move the following sample configuration files from *<application-server-install-dir>/standalone/deployments/cisnatural.war/WEB-INF* to their appropriate location as described below:

   - ***njxnwo-roles.properties* and*njxnwo-users.properties***
     Move these two files to *<application-server-install-dir>/standalone/configuration*.

2. Add the following security domain definition in the file *standalone.xml*, under `<security-domains>`:

```
<security-domain name="NaturalWebIOAndAjaxRealm" cache-type="default">
    <authentication>
        <login-module
            code="UsersRoles"
            flag="required">
            <module-option name="usersProperties"
        value="${jboss.server.config.dir}/njxnwo-users.properties"/>
            <module-option name="rolesProperties"
        value="${jboss.server.config.dir}/njxnwo-roles.properties"/>
            <module-option name="realm" value="NaturalWebIOAndAjaxRealm"/>
            <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
    </authentication>
</security-domain> ↩
```

This sample configuration uses the login module `UsersRoles`. The login module `UsersRoles` expects the role definitions in one file (*njxnwo-roles.properties*) and the user definitions (password and assignment to roles) in another file (*njxnwo-users.properties*). An example user "admin" with the password "adminadmin" and the role "nwoadmin" is defined to begin with.

You can choose and configure a different login module (for example, one that expects the user and role definitions in a database or in an LDAP directory), or you can even write a custom login module.

Configuring the UserDatabaseRealm (Apache Tomcat only)

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:

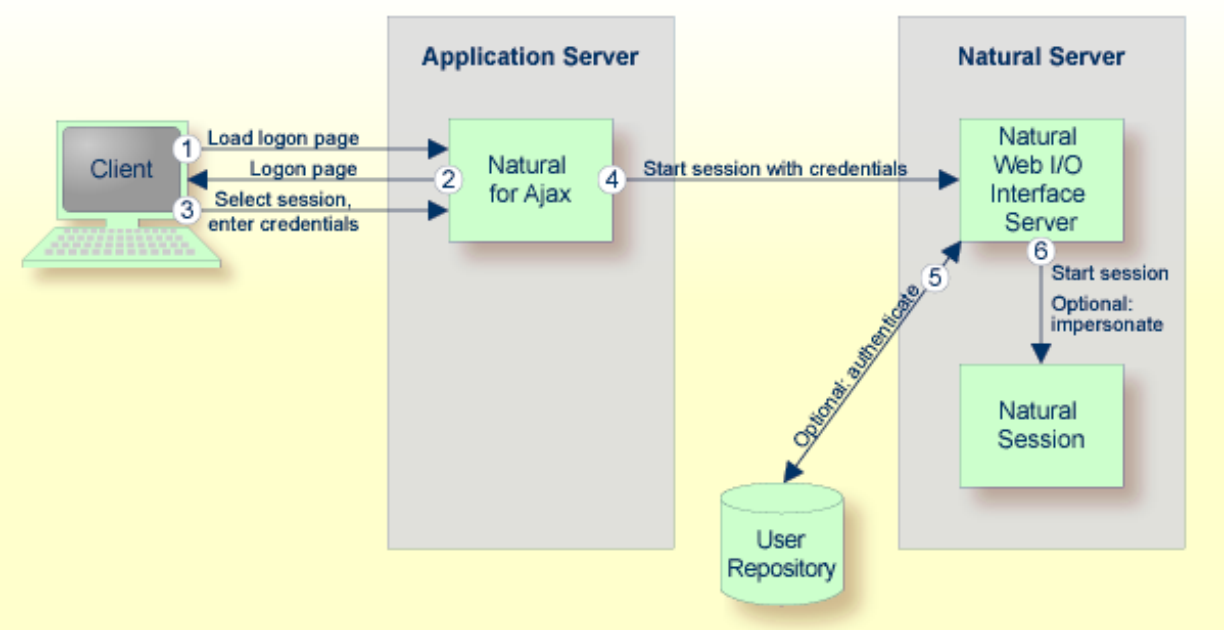
```
<user username="pepe" password="pepe123" roles="nwoadmin"/>
```

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

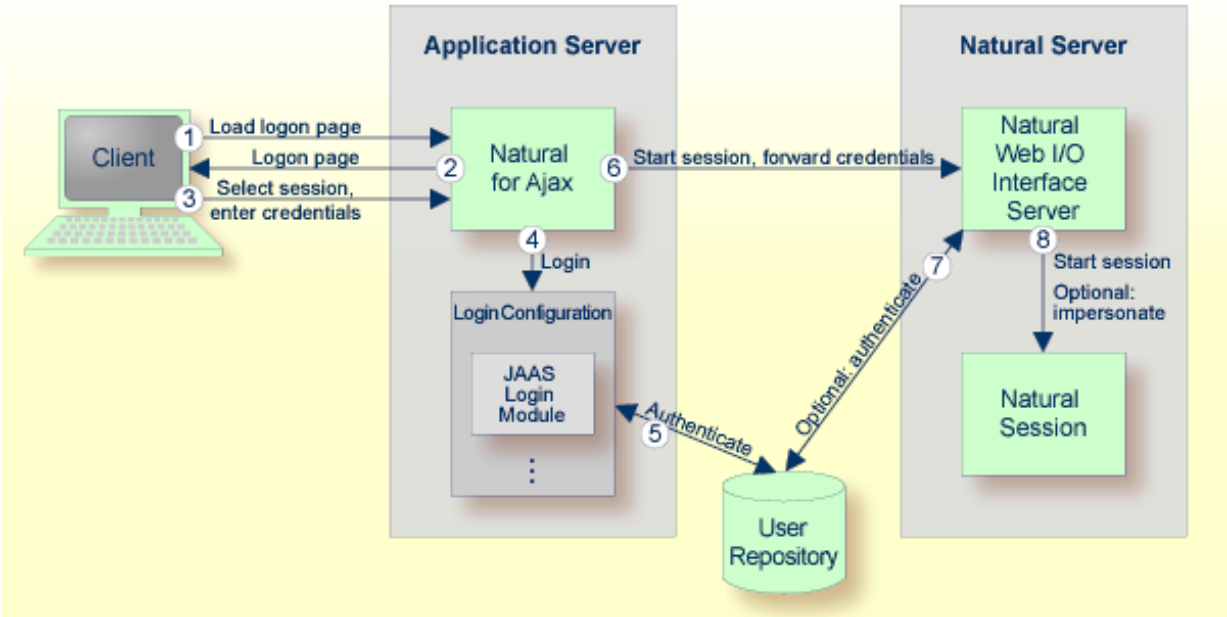
14 Configuring Application-Managed Authentication

General Information

Natural for Ajax is a Java EE-based application that runs on an application server or web container. By default, the access to this application is not secured. The credentials that users enter in the Natural for Ajax logon page (see the graphic below) are used to authenticate them in the selected Natural environment. They are not used to authenticate them on the application server or web container.



You might, however, wish to authenticate the users already in the application server or web container, before they even attempt to access a Natural session. This can be achieved with container-managed security (see *Configuring Container-Managed Security*), but only for a subset of the supported application servers. With application-managed authentication, this can be achieved for all supported application servers and web containers.



When application-managed authentication has been activated, the user is first authenticated on the application server or web container. For the authentication, JAAS-based (Java Authentication and Authorization Service) login modules are used. The login modules to be used and their parameters are configured in a login configuration. The user credentials entered on the Natural for Ajax logon page are authenticated against a user repository that is defined by the configured login modules.

A login configuration can consist of several login modules that are executed one after the other. A specific login module is responsible to forward the entered credentials from the application server or web container to the Natural Web I/O Interface server, so that they can be reused to authenticate the user on the Natural server side.

Activating Application-Managed Authentication

Application-managed authentication is activated on a per-session basis in the **configuration tool**.

To activate-application managed authentication for a session

1 **Invoke** the configuration tool.

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

3 **Add** a new session or **edit** an existing session.

4 Select the **Use JAAS-based authentication** check box.

The **Forward credentials** check box is then selected automatically. This makes sure that the credentials entered for the authentication on the application server or web container are forwarded to the Natural server.

5 Choose the **OK** button.

6 Choose the **Save Configuration** button.

When a user opens the Natural for Ajax logon page and selects the session for which you made the above changes, the credentials entered on the logon page are now used to authenticate the user on the application server or web container and are then forwarded to the Natural server.

This applies also when a user does not explicitly open the logon page in order to select a session manually, but instead passes the session name as an URL parameter to the logon page as described in *Starting a Natural Application with a URL*.

Securing the Logon Page

You may wish to authenticate the users before they even access the Natural for Ajax logon page. This is activated globally in the **configuration tool**.

To secure the logon page

1 **Invoke** the configuration tool.

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

3 Select the **Use secure logon page** check box.

4 Choose the **Save Configuration** button.

When a user opens the Natural for Ajax logon page without specifying a session name beforehand, the user will now be prompted to enter the credentials in order to be authenticated on the application server or web container.

The Login Configuration

With JAAS, authentication is always done against a so-called realm. A realm defines the scope of security definitions. There can be several distinct realms. The user "George" in realm A, for example, is considered to be different from the user "George" in realm B. The realms are usually defined in a login configuration file. The location of this file depends on the application server or web container. A typical realm definition contains a set of login modules that are executed in a specific order to authenticate a user within this realm. The login modules are responsible for the actual authentication.

Natural for Ajax authenticates users against a realm named "NaturalWebIOAndAjaxRealm". Therefore, the login configuration of the application server or web container must contain a realm definition with this name.

Defining the Login Configuration on Wildfly Application Server

The login configuration depends on the Wildfly Application Server version. To define a sample configuration, proceed as described in one of the following sections, depending on the version that you are using:

- *Choosing the Login Module (WildFly)*

Further configuration is described in the version-specific topics below:

- Wildfly Application Server

Wildfly Application Server

If you use other login modules than in the sample configuration, copy the JAR files with these login modules into the *WEB-INF/lib* directory of the Natural for Ajax web application, which is called *cisnatural.war* by default.

In order to prepare for the step *Forwarding the User Credentials to Natural*, you need to provide the Natural for Ajax login module *com.softwareag.njx.loginmodule.NJXLoginModule* in the right place. This login module is contained in the *JBoss7-WildFly8* directory of the installation medium, in the file *njxlogin<nn>.jar*. Copy this file also into the *WEB-INF/lib* directory of of the Natural for Ajax web application.

Defining the Login Configuration on IBM WebSphere Application Server

Copy the JAR files with the login modules to be used into the *lib/ext* directory of your IBM WebSphere installation. The Natural for Ajax login module *com.softwareag.njx.loginmodule.NJXLoginModule* mentioned below is contained in the file *njxlogin<nn>.jar*, which can be found in the WebSphere-specific directory of the installation medium.

To configure the login module

1 Make sure the application server is running.

2 Open your web browser and enter the following URL:

```
http://<host>:<adminport>/ibm/console
```

This opens the Administration Console.

3 Open the tree node **Security > Global security**.

4 On the right side of the screen, open **Java Authentication and Authorization Service**.

5 Choose **Application logins**.

6 Choose **New**.

7 Enter "NaturalWebIOAndAjaxRealm" as an alias.

8 Choose **OK**.

9 Choose **Save**.

10 Select **NaturalWebIOAndAjaxRealm**.

11 Choose **New**.

12 Enter the class name of your login module.

13 Configure the authentication strategy and custom properties of your login module.

14 Choose **OK**.

15 Choose **Save**.

16 Select **NaturalWebIOAndAjaxRealm** once more.

17 Choose **New**.

18 Enter the class name "com.softwareag.njx.loginmodule.NJXLoginModule".

19 Choose **OPTIONAL** as the authentication strategy.

20 Enter "useFirstPass" as the property name.

21 Enter "true" as the property value.

22 Select the **Select** check box.

23 Choose **New**.

24 Enter "storePass" as the property name.

25 Enter "true" as the property value.

26 Select the **Select** check box.

27 Choose **OK**.

28 Choose **Save**.

Defining the Login Configuration on Apache Tomcat

Copy the JAR files with the login modules to be used into the *lib* directory of your Apache Tomcat installation.

The Natural for Ajax login module *com.softwareag.njx.loginmodule.NJXLoginModule* mentioned below is contained in the file *njxlogin<nn>.jar*, which can be found in the Tomcat-specific directory of the installation medium. Copy the file *njxlogin<nn>.jar* into the *WEB-INF/lib* directory of the Natural for Ajax web application, which is called *cisnatural* by default.

In the *conf* directory of your Apache Tomcat installation, add a new properties file named *njx-jaas_config.properties*. Within this file, configure the login modules in the following way:

```
NaturalWebIOAndAjaxRealm {
  your-login-module-class required
    param1="value1"
    param2="value2";

  com.softwareag.njx.loginmodule.NJXLoginModule optional
    useFirstPass=true
    storePass=true;
};
```

On Windows, edit the file *startup.bat* in the Apache Tomcat *bin* directory and add the following line:

```
set JAVA_OPTS=%JAVA_OPTS% ↩
-Djava.security.auth.login.config=%CATALINA_HOME%/conf/njxjaas_config.properties
```

Or, if you have installed Apache Tomcat as a Windows service, specify the above Java option in the Apache Tomcat **Properties** dialog.

On UNIX or Linux, edit the file *startup.sh* in the Apache Tomcat *bin* directory and add the following line:

```
JAVA_OPTS=$JAVA_OPTS ↩
-Djava.security.auth.login.config=$CATALINA_HOME/conf/njxjaas_config.properties
```

Forwarding the User Credentials to Natural

When the user has been authenticated on the application server or web container, the authenticated user with the credentials can be forwarded directly to the Natural Web I/O Interface server. Optionally, the user can be authenticated the Natural Web I/O Interface server again. Also optionally, the started Natural session can be started under the user ID of the client (impersonation).

However, this works only if both the authentication on the application server and the authentication on the Natural Web I/O Interface server are done with the same credentials against the same authentication system. This will be the case, for example, if the Natural Web I/O Interface server is configured to authenticate with RACF and you have configured a login module on the application server or web container that authenticates the user against the same system.

To forward the user credentials to Natural

1 **Invoke** the configuration tool.

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

3 **Add** a new session or **edit** an existing session.

4 Make sure that the **Forward credentials** check box is selected.

5 Choose the **OK** button.

6 Choose the **Save Configuration** button.

Using Software AG Security Infrastructure

Software AG Security Infrastructure (SIN) has originally been designed as a common authentication infrastructure for the webMethods product suite. SIN is based on the above mentioned JAAS (Java Authentication and Authorization Service) framework and provides a number of predefined JAAS-based login modules. For detailed information on these login modules and their configuration, see the Software AG Security Infrastructure documentation at ***http://documentation.software-ag.com/webmethods/security_infrastructure.htm*** (Empower login required).

With this Natural for Ajax distribution, the SIN login modules are delivered in the top level directory */Support/SIN* as a zip file with name *SIN_<version>.zip*. Unzip this file to a temporary directory. The login modules are then contained in the directory with the name *jars*. The JAR files of interest are:

- *sin-common.jar*
- *sin-misc.jar*
- *sin-ssx.jar*

- *sin-xmlserver.jar*

If you want to use the SIN login modules with Natural for Ajax, you have to copy the above JAR files into the proper directory of your application server or web container (for information on the proper directory, see the above sections which describe how to define the login configuration).

Natural for Ajax currently supports the following predefined login modules from the SIN package:

- `SSXLoginModule`
- `SAMLArtifactLoginModule`

SAMLArtifactLoginModule

When this login module is installed and configured for Natural for Ajax, the end user is not prompted for a user name or password during the login.

Make sure that the `opensaml` library and the Apache common libraries are in the class path of the login module (for detailed information, see the description of the `SAMLArtifactLoginModule` in the Software AG Security Infrastructure documentation) and then configure the use of SAML as described below.

To configure the use of SAML

1 **Invoke** the configuration tool.

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

3 Under **Global Settings**, make sure that the **Use SAML for JAAS-based authentication** check box is selected.

4 **Add** a new session or **edit** an existing session.

5 Make sure that the **Use JAAS-based authentication** check box is selected.

When both **Use SAML for JAAS-based authentication** and **Use JAAS-based authentication** are selected, the user name and password fields are not visible in the Natural for Ajax logon page for the used session.

6 Choose the **OK** button.

7 Optional. Under **Global Settings**, select the **Secure logon page** check box.

Now, the secure logon page will only be displayed in the case of an error in the SAML artifact validation.

8 Choose the **Save Configuration** button.

For more information on the above options, see *Session Configuration*.

In both logon pages (Natural for Ajax logon page and secure logon page), the end user is no longer prompted to enter user name and password. Instead, the SAML artefact ticket, which is transported in the HTTP header, is validated. The `SAMLArtifactLoginModule` can extract the user name from the SAML artefact. Therefore, it is not necessary to enter a user name. A valid SAML artefact inside the HTTP header provides the authentication for the Natural for Ajax server.

Important: The Natural for Ajax application does not send a password. Therefore, you have to disable the password check on the Natural Web I/O Interface server. Make sure that only selected Natural for Ajax application servers have access to a Natural server which has been configured in this way.

Using Integrated Authentication Framework (IAF)

With Natural for Ajax, you can authenticate against a Software AG IAF server. Natural for Ajax delivers an IAF login module for this purpose. After you have successfully entered your login credentials, the IAF server creates an IAF token. This token will be sent to Natural for enhanced authentication.

The IAF login module needs a running IAF server. For detailed information on how to set up an IAF server, see the Software AG Security Infrastructure documentation at *http://documentation.softwareag.com/webmethods/security_infrastructure.htm* (Empower login required).

The name of the IAF login module is `com.softwareag.njx.loginmodule.iaf.IAFLoginModule`. It supports the following parameter:

Parameter	Description

`iafConnection`	Mandatory. The address of the IAF server connection. This address must also include the location of the trus
	
	`iafConnection="ssl://myiafserver:11958?TRUST_STORE=C:\\SoftwareAG\\IAF\\iaf\\etc\\IAF`
	`↩`
	
	The default port is 11958.
	
	`TRUST_STORE` must define a valid path on the PC on which the web server is running.

The Natural for Ajax login modules `com.softwareag.njx.loginmodule.NJXLoginModule` and `com.softwareag.njx.loginmodule.iaf.IAFLoginModule` are contained in the following files:

- *njxlogin<nn>.jar*
- *njxiaflogin<nn>.jar*

The above JAR files are contained in the application server-specific directory of the installation medium. You have to copy these files into the proper directory of your application server or web container (see below).

The IAF login module uses parts of the Software AG Security Infrastructure. Therefore, the following additional JAR files are required:

- *IafClient.jar*
- *entirex.jar*
- *sin-common.jar*

The above JAR files are delivered in the top level directory */Support/SIN* of the installation medium, in a zip file with name *SIN_<version>.zip*. Unzip this file to a temporary directory. The JAR files are then contained in the directory with the name *jars*. Note that the file *entirex.jar* is contained in the file *IafClient.jar* and needs to be extracted from there. You have to copy these files into the proper directory of your application server or web container (see below).

Further configuration information is provided in the topics below:

- Configuration on Wildfly Application Server
- Configuration on IBM WebSphere Application Server
- Configuration on Apache Tomcat

Configuration on Wildfly Application Server

Prepare the login modules as described under ***Choosing the Login Module (Wildfly)*** .

Copy all required JAR files (see above) from the installation medium to *standalone\deployments\cisnatural.war\WEB-INF\lib*.

Edit the file *standalone.xml* and add the definition for the IAF login module under `<security-domains>`:

```
<security-domain name="NaturalWebIOAndAjaxRealm" cache-type="default">
 <authentication>
  <login-module code="com.softwareag.njx.loginmodule.iaf.IAFLoginModule" ↩
flag="required">
   <module-option name="iafConnection" ↩
value="ssl://myiafserver:11958?TRUST_STORE=C:\\SoftwareAG\\IAF\\iaf\\etc\\IAFAppCert.jks&amp;VERIFY_SERVER=N"/>
  </login-module>
  <login-module code="com.softwareag.njx.loginmodule.NJXLoginModule" flag="optional">
   <module-option name="password-stacking" value="useFirstPass"/>
   <module-option name="useFirstPass" value="true"/>
  </login-module>
 </authentication>
</security-domain>
```

Configuration on IBM WebSphere Application Server

Prepare the login modules as described above under *Defining the Login Configuration on IBM WebSphere Application Server*.

Copy all required JAR files (see above) from the installation medium to *lib/ext*.

Configure the IAF login module as described below.

To configure the IAF login module

1 Configure the login module in the same way as described under *Defining the Login Configuration on IBM WebSphere Application Server*.

2 For the step "Enter the class name of your login module", enter the following:

```
com.softwareag.njx.loginmodule.iaf.IAFLoginModule
```

3 For the step "Configure the authentication strategy and custom properties of your login module" proceed as follows:

- Enter the class name of your login module: "com.softwareag.njx.loginmodule.iaf.IAFLoginModule".
- Choose **REQUIRED** as the authentication strategy.
- Enter "iafConnection" as the property name.
- Enter the IAF connection string as the property value. Example:

```
iafConnection="ssl://myiafserver:11958?TRUST_STORE=C:\\SoftwareAG\\IAF\\iaf\\etc\\IAFAppCert.jks&VERIFY_SERVER=N";
```

4 Proceed with the remaining configuration steps described under *Defining the Login Configuration on IBM WebSphere Application Server*.

Configuration on Apache Tomcat

Prepare the login modules as described above under *Defining the Login Configuration on Apache Tomcat*.

Copy all required JAR files (see above) from the installation medium to *WEB-INF/lib*.

Edit the file *njxjaas_config.properties* and add the definition for the IAF login module:

```
NaturalWebIOAndAjaxRealm {
  com.softwareag.njx.loginmodule.iaf.IAFLoginModule required
    ↩
iafConnection="ssl://myiafserver:11958?TRUST_STORE=C:\\SoftwareAG\\IAF\\iaf\\etc\\IAFAppCert.jks&VERIFY_SERVER=N";

  com.softwareag.njx.loginmodule.NJXLoginModule optional
    useFirstPass="true"
    tomcatMode="true"
    storePass="true";
};
```

15 Wrapping a Natural for Ajax Application as a Servlet

In a production environment, it is inconvenient to start an application with a URL such as the following:

http://myappserver:4711/cisnatural/servlet/StartCISPage?PAGEURL=%2Fcisnatural%2FNatLogon.html&xciParameters.natserver=mywebio&xciParameters.natprog=nwo.sh&xciParameters.natport=4712&xciParameters.natparam=stack%3D%28logon+SYSEXNJX%3BMENU-NJX%3BFIN%29

The URL can be shortened by defining a corresponding session profile in the **configuration tool**. For example:

http://myappserver:4711/cisnatural/servlet/StartCISPage?PAGEURL=%2Fcisnatural%2FNatLogon.html&xciParameters.natsession=DemoApplication

However, this shortened URL is still not practical for security reasons because end users should not be allowed to access the generic servlet `StartCISPage`.

When you define a dedicated servlet for each application, you can easily define the security constraints for each application in the file *web.xml* (for further information on this file, see ***Configuring Container-Managed Security***). With the servlet `com.softwareag.cis.server.StartCISPageWithParams`, you define a wrapper servlet for a given Natural for Ajax application so that the application can later be invoked with a URL such as the following:

http://myappserver:4711/cisnatural/servlet/DemoApplication

Note: The servlet `com.softwareag.cis.server.StartCISPageWithParams` can also be used with the HTTP method `POST`. See ***Starting a Natural Application with a URL***.

The following example shows how you define an application as a servlet in the file *web.xml*.

```
<servlet id="DemoApplication">
  <servlet-name>DemoApplication</servlet-name>
  <display-name>DemoApplication</display-name>
  <servlet-class>com.softwareag.cis.server.StartCISPageWithParams</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param id="OVERWRITE">
    <param-name>OVERWRITE</param-name>
    <param-value>false</param-value>
  </init-param>
  <init-param id="PAGEURL">
    <param-name>PAGEURL</param-name>
    <param-value>/cisnatural/NatLogon.html</param-value>
  </init-param>
  <init-param id="xciParameters.natsession">
    <param-name>xciParameters.natsession</param-name>
    <param-value>Local</param-value>
  </init-param>
  <init-param id="xciParameters.natserver">
    <param-name>xciParameters.natserver</param-name>
    <param-value>localhost</param-value>
  </init-param>
  <init-param id="xciParameters.natport">
    <param-name>xciParameters.natport</param-name>
    <param-value>2900</param-value>
  </init-param>
  <init-param id="xciParameters.natparamext">
    <param-name>xciParameters.natparamext</param-name>
    <param-value>STACK=(LOGON SYSEXNJX;MENU-NJX;FIN)</param-value>
  </init-param>
</servlet>
```

You can omit the parameters `xciParameters.natserver`, `xciParameters.natport` and `xciParameters.natparamext` if the corresponding values are defined in the session definition that is referenced in `xciParameters.natsession`. This the recommended way, because the settings can thus be changed in the configuration tool at any time without the need to adapt the file *web.xml*.

To complete the definition, you define a corresponding servlet mapping in the file *web.xml*:

```
<servlet-mapping>
  <servlet-name>DemoApplication</servlet-name>
  <url-pattern>/servlet/DemoApplication</url-pattern>
</servlet-mapping>
```

With this servlet mapping, you can now start your application by URL. Example: *http://myhost:myport/mywebapp/servlet/DemoApplication*

If you additionally want to prevent users from starting other NJX applications you can do the following:

Exchange the servlet class `com.softwareag.cis.server.StartCISPage` with a different servlet class, namely `com.softwareag.cis.server.StartCISPageInSession`. This servlet class cannot be called directly; it can only be called in an Application Designer session which is already active. Therefore, each attempt to start an arbitrary - not wrapped - application by just building a URL based on `StartCISPage` will result in an error message.

To exchange the servlet class, you change the following in the file *web.xml*

```
<servlet id="StartCISPage">
  <servlet-name>StartCISPage</servlet-name>
  <display-name>StartCISPage</display-name>
  <servlet-class>com.softwareag.cis.server.StartCISPage</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

to

```
<servlet id="StartCISPage">
  <servlet-name>StartCISPage</servlet-name>
  <display-name>StartCISPage</display-name>
  <servlet-class>com.softwareag.cis.server.StartCISPageInSession</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

If your application uses pop-ups, subcispage controls or web I/O pages you have to add the parameter `ALLOWSUBPAGES` to the servlet definition in your *web.xml*:

```
<servlet id="StartCISPage">
  <servlet-name>StartCISPage</servlet-name>
  <display-name>StartCISPage</display-name>
  <servlet-class>com.softwareag.cis.server.StartCISPageInSession</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param id="ALLOWSUBPAGES">
    <param-name>ALLOWSUBPAGES</param-name>
    <param-value>true</param-value>
  </init-param>
</servlet>
```

For a workplace application, the definition of the `com.softwareag.cis.server.StartCISPageWithParams` in the *web.xml* is slightly different. You do not define the start-up page of the workplace directly in the `PAGEURL` parameter. Instead, you define an intermediate navigation page */HTMLBasedGUI/com.softwareag.cis.util.navigatetopage.html*. The start-up page of the workplace is specified in the `navPage` parameter. The navigation page makes sure that an Applicaton Designer session is created before it navigates to the start-up page of the workplace.

```
<servlet id="WorkplaceDemo">
  <servlet-name>WorkplaceDemo</servlet-name>
  <display-name>WorkplaceDemo</display-name>
  <servlet-class>com.softwareag.cis.server.StartCISPageWithParams</servlet-class>
  <load-on-startup>1</load-on-startup>
  <init-param id="OVERWRITE">
    <param-name>OVERWRITE</param-name>
    <param-value>false</param-value>
  </init-param>
  <init-param id="PAGEURL">
    <param-name>PAGEURL</param-name>
    ↩
<param-value>/HTMLBasedGUI/com.softwareag.cis.util.navigatetopage.html</param-value>
  </init-param>
  <init-param id="navPage">
    <param-name>navPage</param-name>
    <param-value>/njxdemos/wpdynworkplace.html</param-value>
  </init-param>
</servlet>
```

In this case, the corresponding servlet mapping is defined as follows:

```
<servlet-mapping>
  <servlet-name>WorkplaceDemo</servlet-name>
  <url-pattern>/servlet/WorkplaceDemo</url-pattern>
</servlet-mapping>
```

Note: For further information on the above mentioned servlets, see the Java API documentation which is provided with Application Designer.

16 Customizing Error Pages

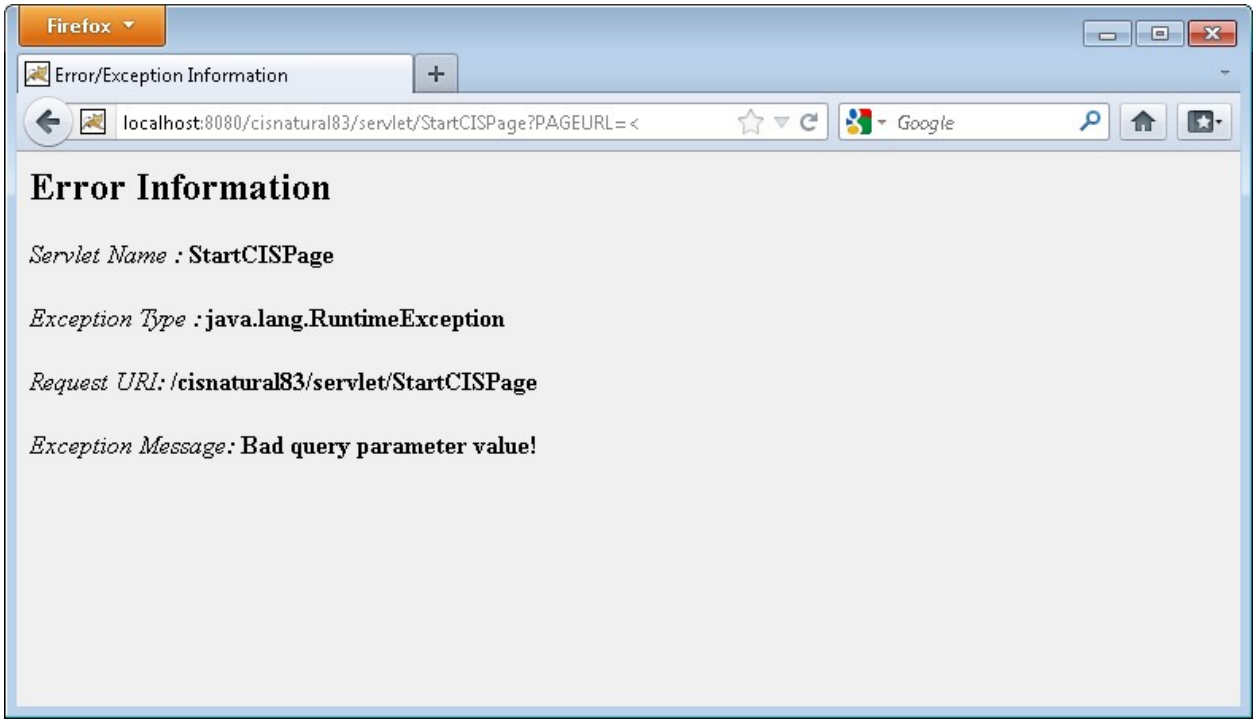
In the case of unexpected errors, most application servers show a default error page. This default error page mostly contains information such as stack traces. Showing full stack traces in a production environment is regarded as a security risk. To avoid this vulnerability, you can configure your own error pages in the *web.xml* file of your web application. For your convenience, the product contains a ready-to-use error handling servlet. The following example shows how to configure this servlet in the *web.xml* file:

```
<servlet id="DefaultErrorHandler">
         <servlet-name>DefaultErrorHandler</servlet-name>
        <servlet-class>com.softwareag.cis.server.DefaultErrorServlet</servlet-class>
</servlet>

<servlet-mapping>
     <servlet-name>DefaultErrorHandler</servlet-name>
     <url-pattern>/DefaultErrorHandler</url-pattern>
</servlet-mapping>

<error-page>
      <exception-type>java.lang.Throwable</exception-type >
      <location>/DefaultErrorHandler</location>
</error-page>
```

The following is a sample error page that has been generated by the `com.softwareag.cis.server.DefaultErrorServlet`:



As an alternative to this default error handling servlet, you can add your own error handling servlets and/or error pages.

17 Configuring SSL

General Information

Trust files are used for a secure connection between the Natural Web I/O Interface server and Natural for Ajax. 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* in *Installing and Configuring the Natural Web I/O Interface Server* which is part of the *Natural Web I/O Interface* documentation for your Natural platform.

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, Natural for Ajax 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, Natural for Ajax 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*.

18 Logging

General Information

Natural for Ajax uses the Java Logging API. In case of problems with Natural for Ajax, 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*. The path to this file depends on the application server. Example for JBoss Application Server:

<application-server-install-dir>/server/default/deploy/njx<nn>ra.rar/log

Logging on JBoss Application Server

JBoss Application Server uses a different logging API (log4j). In this case, we recommend that you enable the file handler in the configuration file *natlogger.xml*.

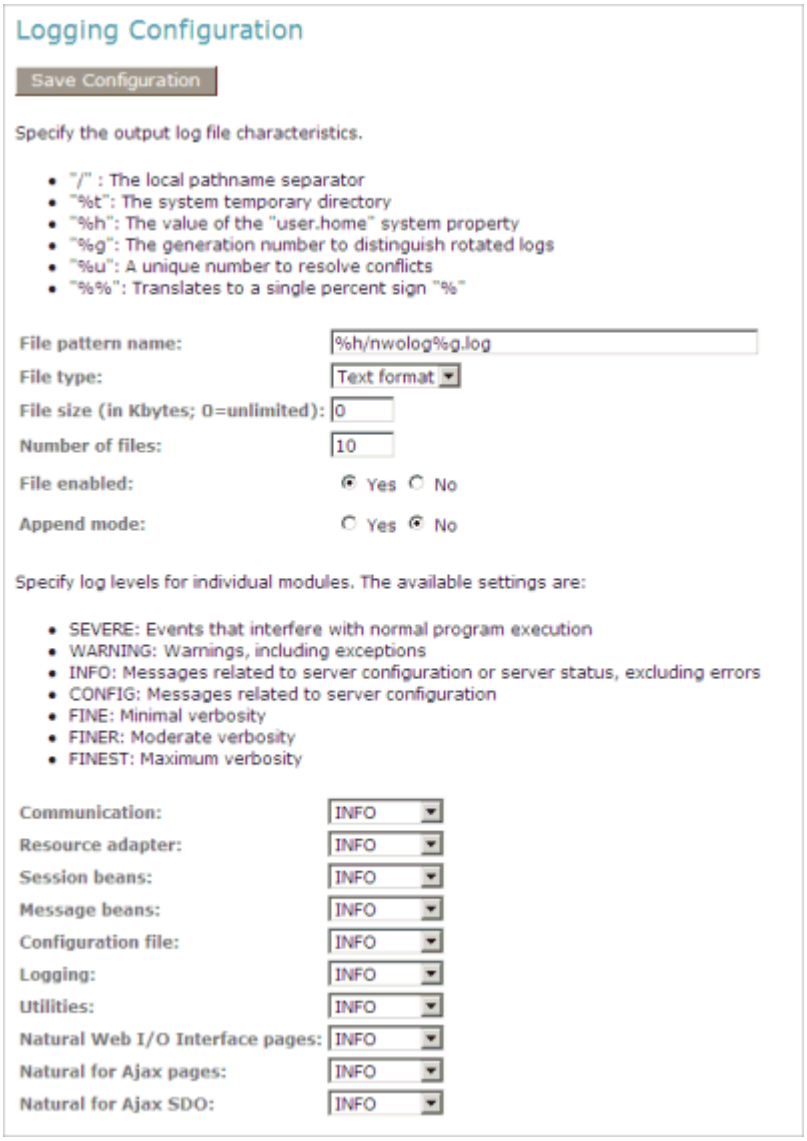
Invoking the Logging Configuration Page

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

To invoke the Logging Configuration page

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

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



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

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

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

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

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

Overview of Options for the Output File

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

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