



NATURAL

Natural

Web Interface

Version 5.1.1 for Windows

Version 3.1.6 for Mainframes

Version 5.1.1 for UNIX and OpenVMS

 **SOFTWARE AG**



This document applies to Natural Version 5.1.1 for Windows, Version 3.1.6 for Mainframes, Version 5.1.1 for UNIX and OpenVMS, and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© June 2002, Software AG
All rights reserved

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

Table of Contents

Natural Web Interface	1
Natural Web Interface	1
Introducing the Natural Web Interface	2
Introducing the Natural Web Interface	2
What is the Natural Web Interface	3
Architecture	3
Communication Using Natural RPC Techniques	3
Communication Using DCOM Techniques	3
Natural Web Interface Modules	4
Features	4
Calling Natural Subprograms from a Web Page	4
Feedback to the User with a Formatted Web Page	5
Proven Middleware	5
Web Page Creation	5
Functionality	6
Security	6
Natural Web Interface Essentials	8
Natural Web Interface Essentials	8
Working with the Natural Web Interface	9
Working with the Natural Web Interface	9
Setting up your Environment	9
Prerequisites on the Web Environment Side	9
Middleware Prerequisites	9
Prerequisites on Natural Server Side	9
Building Subprograms in Natural	9
Before You Write Your Subprograms	10
Ways to Create Your Subprograms	11
General Programming Considerations	19
Changing the Amount of Data Transferred	21
Testing Subprograms	21
The Natural Web Server Extensions	22
Data Areas	22
Naming Conventions of the Library SYSWEB	23
Conversion Program: HTML to Natural	24
Conversion Program: HTML to Natural	24
Using the Conversion Program	24
Functions and PF Keys	25
Generating a Subprogram/Subroutine to be called direct from the Web	25
Inserting a Natural Tag	25
Attributes DATA, LDA, GDA, SUB, NOT	26
Comment Tag	26
ASP-like Script Commands	26
Additional Script Directives	26
Example 1 of a Simple Generation	26
Example 2 of a Simple Generation with a Natural Tag	27
Options	29
Input/Output Fields	30
Functions and PF Keys	31
Generating a DCOM Class	31
Invoking Generate Class	32
Input/Output Fields	32
Example for Library SYSWEB	32
Functions and PF Keys	33

Online Test Utility WEB-ONL	33
Running the Application	33
Input/Output Fields	34
Functions and PF Keys	34
Programming Tips	35
Programming Tips	35
Editing in Lower Case	35
Quote v. Apostrophe	35
Variables defined by Value	35
Access to Resources	36
Constant Values	36
Creating a New Page	36
DCOM / RPC	37
Web Interface Administration	38
Web Interface Administration	38
Set the Size of the Return-Page Transport Buffer	38
Changing the Transport Send Buffer Width	38
Changing the Received Data Buffer Width	38
Changing Your Return Page	38
Set the Size of the Return Page	39
Create a User-Defined Error Page	39
Create a User-Defined Error Page XML-Style	39
Alphanumeric-to-HTML Conversion	39
Alphanumeric-to-URL Conversion	41
Demonstration Application - without JavaScript	42
Demonstration Application - without JavaScript	42
Business Requirements	42
Design Decisions	44
Libraries, Modules and Naming Conventions	44
Starting the Demonstration Application	44
Starting the Natural Web Interface Online Manual	45
Requirements	45
Demonstration Application - with JavaScript	46
Demonstration Application - with JavaScript	46
Business Requirements	46
Design Decisions	48
Starting the Demonstration Application	48
Requirements	48
Natural Web Interface Error Messages	49
Natural Web Interface Error Messages	49
Error Messages	49
Natural Web Interface Installation	50
Natural Web Interface Installation	50
Configuring the Natural Web Interface	51
Configuring the Natural Web Interface	51
Supported HTTP Servers	51
Configuring RPC and RPC Server	51
Natural Version 3.1.5 for Mainframes / Natural Version 4.1.2 for UNIX Server / Natural Version 5.1.1 for Windows	51
EntireX / ENTIRE Broker SDK	52
Configuring the DCOM Server	52
DCOM	52
NaturalX Server	53
Configuring the Web Interface	53
Natural Web Interface	53
Natural Web Server Extensions for RPC	53

Natural Web Server Extensions for DCOM	53
Natural Web Server Extensions for NSAPI	54
Configuring an HTTP Server	55
Communication with Natural Security	55
HTTP Server Security:	55
EntireX Security:	55
Natural Security:	55
Web Interface Troubleshooting	56
Web Interface Troubleshooting	56
Natural Web Interface Programming Guide	58
Natural Web Interface Programming Guide	58
Example Programs	58
Web Interface Specification	59
Web Interface Specification	59
Demonstration Application	59
Main Menu	59
Listing	59
Browse	60
Select	60
New	60
Show	60
Change	60
Delete	60
Layout	60
Web Interface HTML Design	62
Web Interface HTML Design	62
The Given Layout as HTML Table	62
HTML Source of the Table	63
Restrictions	64
Colors	64
HTML 4.0 and Cascading Style Sheets	65
Fixed Table Size	65
Chosen Layout	66
Input Layout	66
Functional Parts	71
How to Use Frames	71
How to Use JavaScript	71
Web Interface Program Design	72
Web Interface Program Design	72
Data Input	72
Global Data	72
1. Cookies	72
2. Hidden Input Fields and Additional URL Parameters	73
3. Data Saved on the Server Side (in a Database)	73
Dispatch	73
External Data	74
Modular Pages	75
Chosen Program Design	76
Functional Parts	76
Web Interface Implementation	79
Web Interface Implementation	79
HTML to Natural	79
Reuse of Global Parts	79
Use of w3text versus w3html	79
Increased Performance using w3text/w3html	79

Web Interface Fine Tuning	80
Web Interface Fine Tuning	80
Entrance Tunnel	80
Natural Web Server Extensions - Overview	83
Natural Web Server Extensions - Overview	83
Natural Web Server Extensions - Introduction	84
Natural Web Server Extensions - Introduction	84
General Information	84
Installation - RPC / DCOM	84
Transformations	85
Variables	85
Error Logging and Messages	85
Calling Programs	85
Natural Web Server Extensions - Initialization File	87
Natural Web Server Extensions - Initialization File	87
General Information	87
RPC Parameters	87
DCOM Parameters	88
Natural Web Server Extension Settings	88
HTTP Server Variables	91
Additional Variables	91
Error Templates	92
Default Error Report	92
Specifying your own Error Template	93
Example of an Error Template	94
Natural Web Server Extensions - Error Messages	95
Natural Web Server Extensions - Error Messages	95

Natural Web Interface

The Natural Web Interface is a link between a Web Server (more precisely: HTTP server) and your Natural environment.

The Natural Web Interface documentation comprises the following documents:

- [Introducing the Natural Web Interface](#)
- [Natural Web Interface Installation](#)
- [Natural Web Interface Essentials](#)
- [Natural Web Interface Programming Guide](#)
- [Natural Web Interface Configuration](#)

Introducing the Natural Web Interface

More and more organizations need to offer information or services via the Internet. Gone are the days where static HTML pages were sufficient for the daily visitors to a web page. Today, increasingly sophisticated HTML pages are competing in the web, and the demand for full access to business logic via the Internet is increasing tremendously. The database management systems containing business-critical information are mostly based on heavy-duty servers like mainframes.

This section covers the following topics:

- What is the Natural Web Interface
 - Architecture
 - Natural Web Interface Modules
 - Features
 - Functionality
 - Security
-

What is the Natural Web Interface

The Natural Web Interface is a link between a Web Server (more precisely: HTTP server) and your Natural environment. This can be on a separate server machine (such as a mainframe) or on the same machine as the HTTP server (e.g. Netscape's Communication Server or Microsoft's IIS).

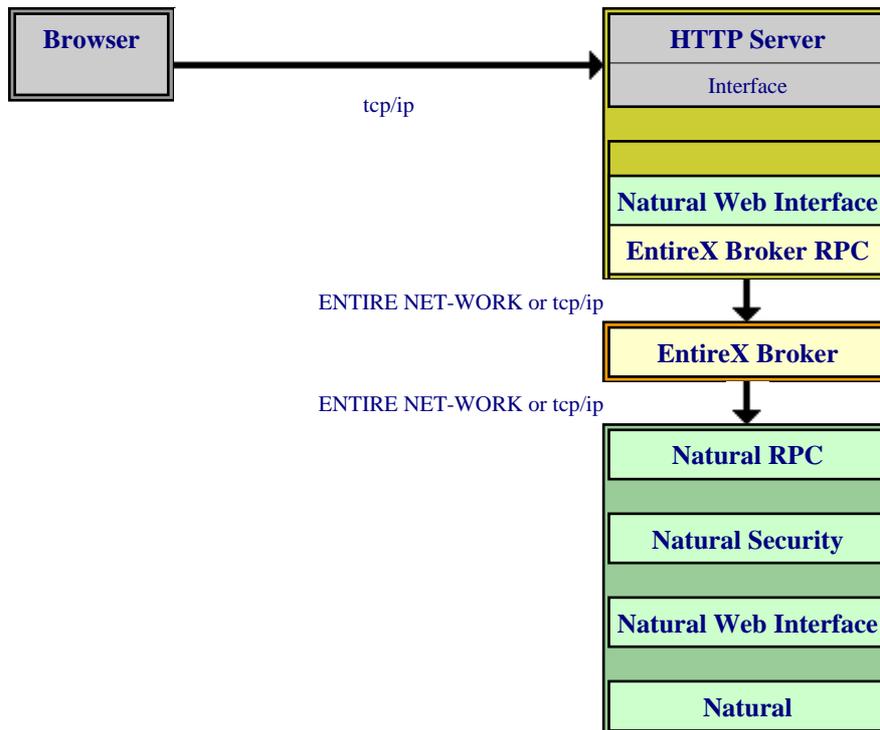
Contents of web pages can easily be created dynamically by a Natural program. This is a basis for implementing a real interactive application on the web.

An interactive application enables users to input information and react by issuing output depending on that input. Examples of Web-based applications are order entry systems, travel booking services and parcel tracking systems. This considerably increases the scope of Natural applications. Not just in-house users, but also potential customers all over the world can now use the same application.

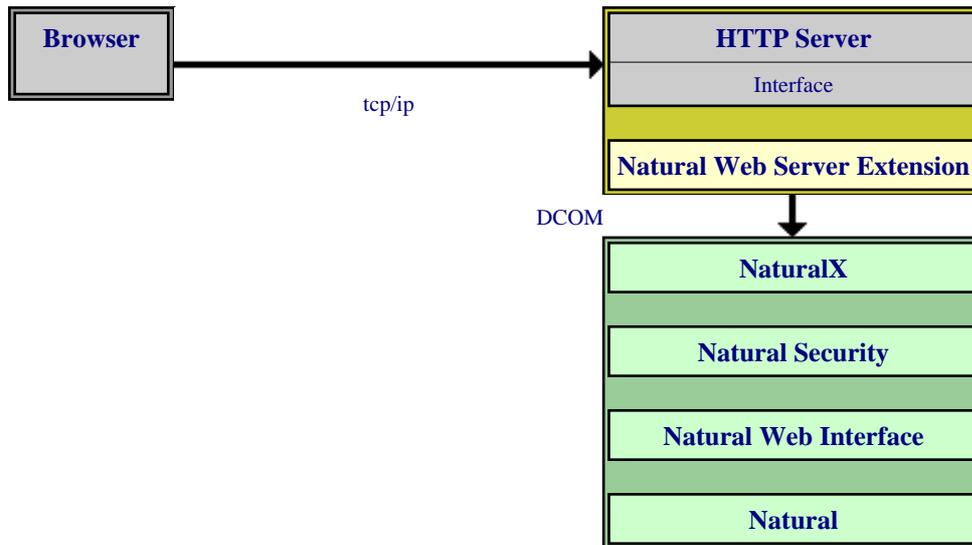
And best of all: to implement such an application, Natural users do not have to learn a new programming language. Navigation and user input/output are implemented fully in Natural (with some additional embedded HTML statements).

Architecture

Communication Using Natural RPC Techniques



Communication Using DCOM Techniques



Natural Web Interface Modules

The Natural Web Interface comprises three internal modules:

1. **Natural Web Interface** - the HTML API and the HTTP API of Natural
2. **Natural Web Server Extensions** - the part which provides the interface to the web server on the same machine
3. **Necessary middleware**: EntireX or Entire Broker using RPC or DCOM technology

Features

Calling Natural Subprograms from a Web Page

One of the main features of the Natural Web Interface is that Natural subprograms can be called from a web page. This can be done using forms on a web page that contains input fields and buttons. Users can enter data and submit these data by clicking one of the buttons. This executes a Natural subprogram which passes the user data as parameters.

This allows easy access to application functions (= subprograms). Simple database access for retrieving data using SQL (and an ODBC driver) as offered by most Web Servers is not enough for implementing an interactive application. You also need business logic to ensure data consistency and processing of the user data.

Business logic such as consistency and plausibility checks usually already exist, as they were implemented for operational applications in the past. If they were implemented as separate Natural modules (such as subprograms, programs, or subroutines) they can easily be re-used and do not have to be re-implemented in a different environment or different language.

Therefore, no special interface program has to be written to connect the web server with the business functions. The Natural Web Interface is a standardized interface for that purpose.

No programming language has to be learned and existing skills can be leveraged (except for HTML statements to format the output pages).

Feedback to the User with a Formatted Web Page

The second important part of an interactive application on the web is the feedback to the user with formatted web pages. With Natural Web Interface these web pages can be formatted dynamically according to the application's needs.

A benefit is that the control of layout and contents of these pages is fully at the application/program level, not outside in separate directories.

And also: as Natural can gather data and information from a wide variety of sources (Adabas, RDBMSs, VSAM, sequential files, even system information with Natural Process) the type of application is virtually unlimited - any application you can build with Natural you can integrate with the web).

Proven Middleware

The Natural Web Interface is based on the proven set of middleware products from Software AG: the Entire product family.

This allows seamless integration in an existing client/server environment. The web connection is just another client, which can be connected to existing Natural servers. If Entire Net-Work is installed, you do not need to install another set of middleware products.

Since Natural for Windows and UNIX/OpenVMS Version 4.1, the interface can call Natural DCOM classes. The methods called, with a specific interface, can map to the same subroutines used through remote procedure call (RPC).

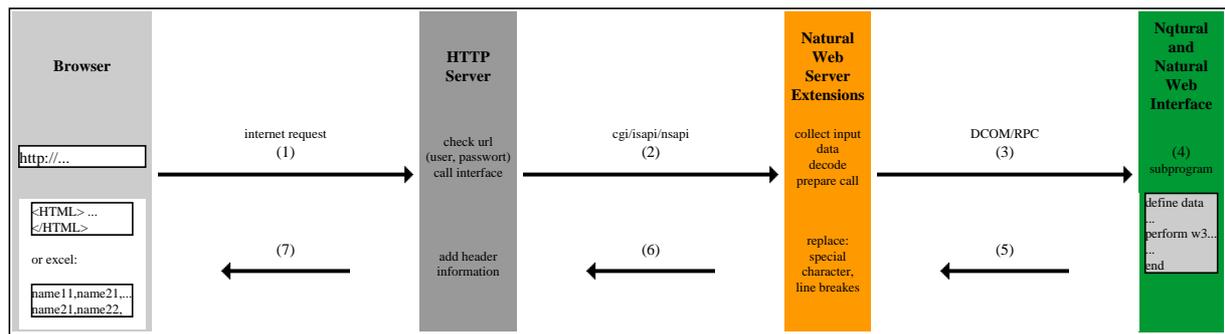
Web Page Creation

Web pages can be created with standard tools (e.g. FrontPage) or with the web page creation tool using the Natural generation functionality. From the Natural server, subprograms can be generated. There's no need to acknowledge any other programming language or web-page creation tool.

Functionality

Requests from a web page in the user's browser are passed to the web (or HTTP) server. Provided that this was a form requesting execution of a Natural subprogram, this request is then passed to the Natural Web Server Extensions part which executes the Natural subprogram via EntireX RPC or DCOM. The program takes any user data as parameters and then issues a set of programs to provide the feedback to the user.

The following diagram illustrates how the Natural subprograms are called from an HTML browser. Each stage of the process is identified by a number; what happens at these stages is explained below.



1. HTML Browser Requests URL.
Your browser requests a URL identifying the program you want to call on the server side.
2. Web Server calls the Natural Web Server Extension CGI.
The web server takes the URL and calls Natural Web Server Extensions.
3. Natural Web Server Extension converts the call to RPC.
The Natural Web Server Extension program "translates" the URL into a Natural RPC that invokes the Natural server program originally identified by the URL.
4. Natural subprogram is executed and generates a return page.
The Natural subprogram on the server is executed and generates an HTML return page.
5. Return Page is sent back to the Natural Web Server Extension.
The HTML return page is sent back as response of the subroutine call.
6. Natural Web Server Extension sends back the return page to the Web Server.
The Web Server adds header information and sends it to the browser.
7. The browser receives the answer to what it was sent out as a request for an URL.

Note:

In the context of the Natural Web Interface, only external subroutines can return output.

Security

Pages called via Natural Web Interface can work together with Natural Security. This is accomplished as follows:

- First your Natural Web Server Extension has to be defined as restricted page at your HTTP server.
- If this is done, you will be prompted for user ID and password by your browser if you request an Adapter page.
- The HTTP server will now verify the given data with its database.
- If the user is authorized, Natural Web Server Extension is called with the remote user's name.
- If the Natural RPC server is started with Natural Security, the given name will be set as *USER.
- As an authentication is already done by the HTTP server, no password checking will be done on the Natural side. Therefore, the Natural RPC server has to be started with AUTO=ON.

A second scenario is that, at the initialization file, a specific, fixed, defined user ID and password is set to communicate with a Natural RPC server with Natural Security. See also Communication with Natural Security

Natural Web Interface Essentials

This part of the Natural Web Interface documentation describes how the Natural Web Interface enables you to create web-enabled Natural subprograms and how a web browser can call these subprograms and can receive a page in return.

This part of the documentation also outlines those functions of the Software AG product EntireX which are relevant to the operation of the Natural Web Interface. For more information, see the EntireX documentation.

You should know the essentials of HTML, of web browsers and of the environments in which the web browsers operate. You should also have a sound knowledge of Natural in a client-server environment.

This part of the Natural Web Interface documentation contains the following sections:

- | | |
|--|--|
| ● Working with the Natural Web Interface | Describes how to set up the environment and how to work with subprograms. |
| ● Natural Web Server Extensions | Describes how the Natural Web Interface enables you to create web-enabled Natural subprograms and how a web browser can call these subprograms and can receive a page in return. |
| ● Conversion Program HTML to Natural | Describes how to use the HTML to Natural conversion programs. |
| ● Tips on Programming | Contains tips for the usage of the Natural Web Interface to enable you to build better web programs. |
| ● Administration | Describes how to set formats, how to define error pages, how to convert to HTML and to decode an URL. |
| ● Demonstration Application without JavaScript | Contains a demonstration application which shows the use and programming of the Natural Web Interface. |
| ● Demonstration Application with JavaScript | Contains a more comprehensive demonstration application. This demonstration application requires a browser which supports Java. |
| ● Natural Web Interface Error Messages | Contains a list of error messages you may receive when you are working with the Natural Web Interface. |

The Natural library SYSWEB contains all modules of the Natural Web Interface.

Working with the Natural Web Interface

This section covers the following topics:

- Setting up your Environment
 - Building Subprograms in Natural
-

Setting up your Environment

Prerequisites on the Web Environment Side

The following software must be installed:

On the web client:	Browser software, such as Netscape Navigator or Microsoft Internet Explorer.
On the web server:	HTTP server software, such as Netscape Fast Track Server, Microsoft Internet Information Server or Apache Server.

Middleware Prerequisites

Different prerequisites must be met if communication is to be used by RPC or DCOM:

RPC	The broker of the Software AG product EntireX must be installed (for installation information, see EntireX documentation).
DCOM	On non-Windows platforms, DCOM is not a part of the operating system. The Software AG product EntireX contains a DCOM implementation for several platforms and must be installed (for installation information, see EntireX documentation).

The Natural Web Server Extensions part is needed for communication between a web browser and a Natural DCOM/RPC server.

Prerequisites on Natural Server Side

The following prerequisites must be met:

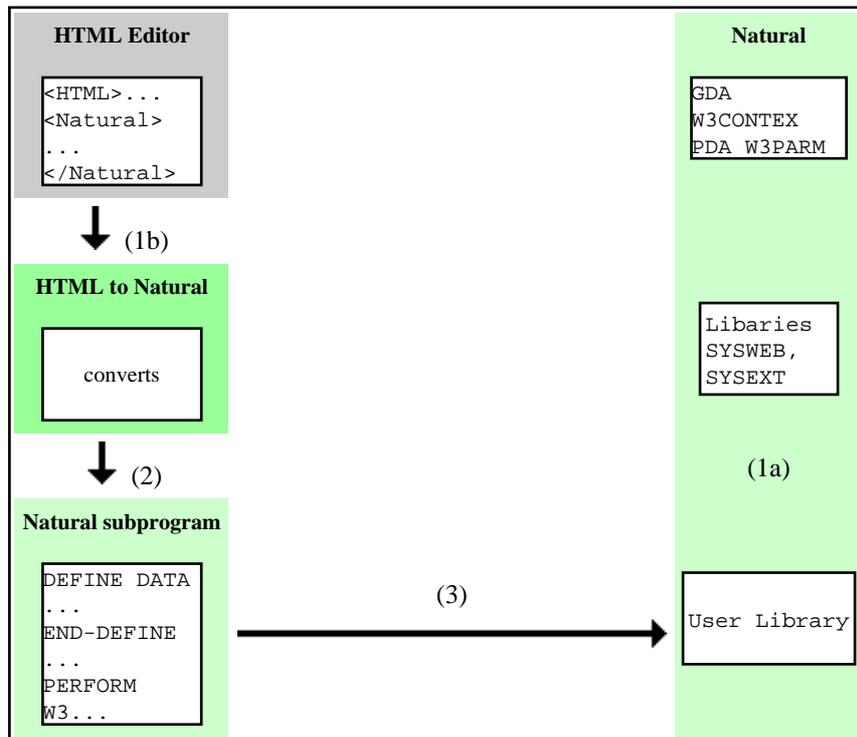
- Natural Version 2.3.1/3.1.1 (Mainframe), 4.1.1 (UNIX / NT) or above must be installed.
- The library SYSWEB (the Natural Web Interface's server programs and server-program online documentation).
Either Natural steplibs must be available or the contents of the library SYSWEB must be copied to the library SYSTEM or to the user library that will be called by the RPC.
- The global data area W3CONTEX.
- The parameter data Area W3PARM.
- The Natural RPC stub or NaturalX.

Building Subprograms in Natural

The following diagram illustrates how you can build a subprogram:

1. Using a HTML editor
2. You use a HTML editor to enter HTML and Natural code
3. Then convert it to Natural source.
4. Finally move the generated program to Natural. (You code directly in Natural.)

Each stage of the process is identified by a number; what happens at these stages is explained below.



1. 1a. Natural Code is written and stored in User Library.
 You write Natural code on the server side either by including HTML tags in the code or by calling pre-fabricated subprograms that generate HTML tags. Then you store it as a server program or use the subprogram WEB-WIZ to generate a default program.
- 1b. Natural Code is entered as HTML. Continue with 2.
 You use an HTML editor to create HTML pages.
2. Program HTML2NAT generates Natural Sources out of HTML.
 You start the program HTML2NAT out of the library SYSWEB and let it convert your HTML pages created in step 1b.
3. Generated Natural Source is moved to the User Library.
 You move the generated Natural sources to your server environment by using the Natural SYSUNLD function.

Before You Write Your Subprograms

Keep the following things in mind:

- The returning HTML page is limited to the maximum data that can be transmitted. This maximum is determined by the return page variable.
- You must initialize and end the access to the Natural server subroutines by calling the subroutines W3INIT and W3END in the library SYSWEB.
- Always use the parameter data areas W3PARM and W3CONST.
- Use the subprogram WEB-WIZ to generate a frame (default program) for your own program.

Ways to Create Your Subprograms

There are two basic alternatives:

1. Coding in Natural direct
2. Using an HTML editor.

1. Coding In Natural Direct

There are two alternatives:

- Entering calls to SYSWEB subroutines (such as W3HTML or W3TEXT) for your return page in the program editor (see the programs in the library SYSWEB, which help you perform only basic system functions; this approach requires a good knowledge of the data type you are creating, for example HTML or XML); or
- calling subprograms that generate HTML tags (see the library SYSWEB; the programs in the library SYSWEB enable you to perform basic system functions and in addition, the programs in the library SYSWEB generate HTML tags; this approach requires less explicit HTML knowledge and you can still modify the programs you are calling).

Example: Entering Calls to SYSWEB Subroutines in the Program Editor

```

*
* Example E3END
*
DEFINE DATA
PARAMETER USING W3PARAM
LOCAL USING W3CONST
LOCAL
1 W3VALUE          (A250)
END-DEFINE
* --- ERROR HANDLING ---
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
*
* --- INITIALIZE W3 PROCESSING ---
PERFORM W3INIT ##RPC
*
* --- SET TYPE OF RETURN-PAGE ---
PERFORM W3CONTENT-TYPE 'text/html'
* --- WRITE THE DOCUMENT ---
PERFORM W3TEXT '<HTML><BODY><H2>Initialize</H2>'
*
* --- END THE HTML PAGE ---
COMPRESS '<HR>generated:' *DATE *TIME ##HTTP_NEWLINE
        '</BODY></HTML>' ##HTTP_END INTO W3VALUE
PERFORM W3TEXT W3VALUE
*
* --- END W3 PROCESSING ---
PERFORM W3END ##RPC
*
END

```

Example: Calling Subprograms that Generate HTML Tags

```

*
* Example E3IMAGE
*
DEFINE DATA
PARAMETER USING W3PARAM
LOCAL USING W3CONST
LOCAL
1 H3VALUE          (A250)
1 H3VALUE-MAX      (I004)
1 H3URL            (A250)
*
1 II              (I001)
1 GIF             (A064)
END-DEFINE
* --- ERROR HANDLING ---
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
*
* --- INITIALIZE W3 PROCESSING ---
PERFORM W3INIT ##RPC
*

```

```

* --- Pathname of picture ---
PERFORM W3READ-ENVIRONMENT "PICTURES" ' ' H3VALUE H3VALUE-MAX
IF H3VALUE-MAX EQ 0 THEN
  GIF := "/pictures"
ELSE
  GIF := H3VALUE
END-IF
*
* --- START HTML API ---
PERFORM H3-OPEN-HTML 'HTML Api -Image' " " " "
* --- THE LEVEL 2 HEADER ---
PERFORM H3-HEADER 2 'Image'
*
PERFORM H3-RULE 0
*
PERFORM H3-HEADER 4 'left:'
*
COMPRESS GIF '/natw_sam.gif' INTO H3URL LEAVING NO
PERFORM H3-IMAGE H3URL 'NATweb left' 219 229 "L"
*
FOR II 1 TO 10
  PERFORM H3-LINE-BREAK
END-FOR
PERFORM H3-RULE 80
*
PERFORM H3-HEADER 4 'small right:'
*
COMPRESS GIF '/natw_sam.gif' INTO H3URL LEAVING NO
PERFORM H3-IMAGE H3URL 'NATweb small right' 100 100 'R'
*
FOR II 1 TO 5
  PERFORM H3-LINE-BREAK
END-FOR
*
PERFORM H3-RULE 0
*
PERFORM H3-TIME_DATE
*
* --- END HTML API ---
PERFORM H3-CLOSE-HTML
* --- END W3 PROCESSING ---
PERFORM W3END ##RPC
*
END

```

2. Using an HTML Editor

There are two alternatives:

- Creating static pages (you only enter HTML, which will be converted to a Natural subprogram)
- Creating dynamic pages (you enter HTML plus Natural program code).

You can, of course, also create pages that are partly dynamic, partly static.

Example: Creating Static Pages

```
<HTML>
<TITLE>NATweb - Test</TITLE>
<BODY bgColor=d3d3d3 >
<BR>
<center>
<h2>
This Natural subprogram was generated by a HTML page.
</h2>
</CENTER>
</BODY></HTML>
```

This Natural subprogram will be generated from the above HTML page:

```

* ----- SUBPROGRAM generated out of file:
* ----- C:\static.htm
DEFINE DATA
PARAMETER USING W3PARAM
LOCAL USING W3CONST
LOCAL
* ----- PRIVATE VARIABLES -----
1 W3VALUE          (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALISE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
PERFORM W3TEXTLINE '<HTML>'
PERFORM W3TEXTLINE '<TITLE>NATweb - Test</TITLE>'
PERFORM W3TEXTLINE '<BODY bgColor=d3d3d3 >'
PERFORM W3TEXTLINE '<BR>'
PERFORM W3TEXTLINE '<center>'
PERFORM W3TEXTLINE '<h2>'
PERFORM W3TEXTLINE 'This Natural subprogram was generated by a HTML page.'
PERFORM W3TEXTLINE '</h2>'
PERFORM W3TEXTLINE '</CENTER>'
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
* ----- SUBROUTINES -----
*
END

```

Example: Creating Dynamic Pages

```

<Natural><!--
*
* Read form Pers-View starting with value given by the
* Parameter START
*
* Use HTML2NAT to generate a Natural Program
*
* 11.03.02
*
--></Natural>
<!-- Variables to read the environment --->
<Natural data><!--
* ----- DATA -----
1 H3VALUE          (A250)
1 H3MAX            (I4)
--></Natural>
<!-- Head of the HTML page --->
<HTML>
<TITLE>Natural - Environment Test</TITLE>

```

```

<BODY bgColor=d3d3d3 >
<BR>
<center>
<h2>
This Natural subprogram was generated by a HTML page. The program had been
precompiled out of a HTML page.
<br><br>
</h2>
</center>
<br>
<hr>
<!-- Subprogram to write the output to work file,
      from where the server will read it --- >
<Natural DATA><!--
1 #CONTENT (A1/1:48)
1 REDEFINE #CONTENT
  2 #PERSONNEL-NUMBER (N8)
  2 FILLER 1X
  2 #NAME (A20)
  2 FILLER 1X
  2 #FIRST-NAME (A15)
  2 FILLER 1X
  2 #AGE (N2)
--></Natural>
<Natural SUB><!--
* ----- Do the OUTPUT -----
DEFINE SUBROUTINE WRITELINE
  PERFORM W3TEXT "<LI>"
*
  #PERSONNEL-NUMBER:=PERSONNEL-NUMBER
  #NAME:=NAME
  #FIRST-NAME:=FIRST-NAME
  #AGE:=AGE
  PERFORM W3HTMLARRAY #CONTENT(*) 48
*
  PERFORM H3-LINE-BREAK
END-SUBROUTINE
--></Natural>
<UL><PRE>
<!-- Parameter used for reading data from the DATABASE --->
<Natural DATA><!--
* ----- DATA -----
1 #VALUE (A20)
1 PERS-VIEW VIEW OF PERSONNEL
  2 PERSONNEL-NUMBER
  2 NAME
  2 FIRST-NAME
  2 AGE
--></Natural>
<!-- Main program to read the data --->
<Natural NOT>
<LI>Value1
<LI>Value2
<LI>...
</Natural>
<Natural><!--
* --- READ ENVIRONMENT ---
PERFORM W3READ-ENVIRONMENT 'START' 'P' H3VALUE H3MAX
IF H3MAX GT 0 THEN
  #VALUE := H3VALUE
ELSE
  #VALUE := "A"

```

```

END-IF
*
* ----- MAIN -----
F. FIND (100) PERS-VIEW NAME > #VALUE
  IF NO
    COMPRESS 'Sorry nothing found for:' #value '!' INTO H3VALUE
    PERFORM W3HTMLLINE H3VALUE
  END-NOREC
  IF *NUMBER > 0
    PERFORM WRITELINE
  END-IF
END-FIND
*
IF *NUMBER(F.) > 0
  PERFORM H3-RULE 0
*
  COMPRESS 'well done for: ' #value '!' ##HTTP_END INTO H3VALUE
  PERFORM W3HTMLLINE H3VALUE
END-IF
--></Natural>
</PRE></UL>
<! --- The footer of the HTML page --- >
<hr>
<BR>
<center>
<A HREF="index.htm">back to Index</A>
This program has been generated.
<Natural><!--
PERFORM H3-TIME_DATE
--></Natural>
</P>
</CENTER>
</BODY></HTML>

```

This Natural subprogram will be generated from the above HTML page:

```

* ----- SUBPROGRAM generated out of file:
* ----- C:\doit.htm
DEFINE DATA
PARAMETER USING W3PARAM
LOCAL USING W3CONST
LOCAL
* ----- DATA -----
1 H3VALUE          (A250)
1 H3MAX            (I4)
1 #CONTENT (A1/1:48)
1 REDEFINE #CONTENT
  2 #PERSONNEL-NUMBER (N8)
  2 FILLER 1X
  2 #NAME            (A20)
  2 FILLER 1X
  2 #FIRST-NAME     (A15)
  2 FILLER 1X
  2 #AGE            (N2)
* ----- DATA -----
1 #VALUE (A20)
1 PERS-VIEW VIEW OF PERSONNEL
  2 PERSONNEL-NUMBER
  2 NAME
  2 FIRST-NAME
  2 AGE
* ----- PRIVATE VARIABLES -----

```

```

1 W3VALUE          (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALISE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
* ----- MAIN PROGRAM -----
*
* Read form Pers-View starting with value given by the
* Parameter START
*
* Use HTML2NAT to generate a Natural Program
*
* 11.03.2002
*
PERFORM W3TEXTLINE '<! --- Variables to read the environment --->'
PERFORM W3TEXTLINE '<! --- Head of the HTML page --->'
PERFORM W3TEXTLINE '<HTML>'
PERFORM W3TEXTLINE '<TITLE>Natural - Environment Test</TITLE>'
PERFORM W3TEXTLINE '<BODY bgColor=d3d3d3 >'
PERFORM W3TEXTLINE '<BR>'
PERFORM W3TEXTLINE '<center>'
PERFORM W3TEXTLINE '<h2>'
PERFORM W3TEXTLINE 'This Natural subprogram was generated by a HTML page. Th'
  -'e program had been'
PERFORM W3TEXTLINE 'precompiled out of a HTML page.'
PERFORM W3TEXTLINE '<br><br>'
PERFORM W3TEXTLINE '</h2>'
PERFORM W3TEXTLINE '</center>'
PERFORM W3TEXTLINE '<br>'
PERFORM W3TEXTLINE '<hr>'
PERFORM W3TEXTLINE '<! --- Subprogram to write the output to work file'
PERFORM W3TEXTLINE '      from where the server will read it --- >'
PERFORM W3TEXTLINE '<PRE>'
PERFORM W3TEXTLINE '<! --- Parameter used for reading data from the'
  -' DATABASE --->'
PERFORM W3TEXTLINE '<! --- Main Program to read the data --->'
* --- READ ENVIRONMENT ---
PERFORM W3READ-ENVIRONMENT 'START' 'P' H3VALUE H3MAX
IF H3MAX GT 0 THEN
  #VALUE := H3VALUE
ELSE
  #VALUE := "A"
END-IF
*
* ----- MAIN -----
F. FIND (100) PERS-VIEW NAME > #VALUE
IF NO
  COMPRESS 'Sorry nothing found for:' #value '!' INTO H3VALUE
  PERFORM W3HTMLLINE H3VALUE
END-NOREC
IF *NUMBER > 0
  PERFORM WRITELINE
END-IF

```

```

END-FIND
*
IF *NUMBER(F.) > 0
  PERFORM H3-RULE 0
*
  COMPRESS 'well done for: ' #value '!' ##HTTP_END INTO H3VALUE
  PERFORM W3HTMLLINE H3VALUE
END-IF
PERFORM W3TEXTLINE '</PRE>'
PERFORM W3TEXTLINE '<! --- The footer of the HTML page --- >'
PERFORM W3TEXTLINE '<hr>'
PERFORM W3TEXTLINE '<BR>'
PERFORM W3TEXTLINE '<center>'
PERFORM W3TEXTLINE '<A HREF="index.htm">back to Index</A>'
PERFORM W3HTMLLINE 'This program has been generated.'
perform H3-TIME_DATE
PERFORM W3TEXTLINE '</P>'
PERFORM W3TEXTLINE '</CENTER>'
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
*
* ----- SUBROUTINES -----
* ----- Do the OUTPUT -----
DEFINE SUBROUTINE WRITELINE
  PERFORM W3TEXT "<LI>"
*
  #PERSONNEL-NUMBER:=PERSONNEL-NUMBER
  #NAME:=NAME
  #FIRST-NAME:=FIRST-NAME
  #AGE:=AGE
  PERFORM W3HTMLARRAY #CONTENT(*) 48
*
  PERFORM H3-LINE-BREAK
END-SUBROUTINE
END

```

General Programming Considerations

Constant Values in the Local Data Area W3CONST

The local data area W3CONST contains a number of constant values which you might find useful:

##HTTP_NEWLINE, ##HTTP_NEWLINE_LENGTH and ##HTTP_NEWLINE_END

If you enter the ##HTTP_NEWLINE string into your HTML, you can use the subroutines W3TEXT and W3TEXT-ARRAY in the library SYSWEB to create a physical new line by compressing ##HTTP_NEWLINE into the string. If you do not wish trailing blanks to be stripped, you can use ##HTTP_NEWLINE_END. ##HTTP_NEWLINE_LENGTH returns the length of the ##HTTP_NEWLINE string. ##HTTP_NEWLINE_END is always one character longer.

##HTTP_END

If you enter this string into your HTML, all routines which write their output to the return page will remove trailing blanks. If you do not wish trailing blanks to be removed, compress a ##HTTP_END string at the end of your string. There will be more output processing as a result.

##W3ERROR

Parameter used for calling W3ERROR.

##HTML_LT

Constant HTML value for "less than" sign (<).

##HTML_GT

Constant HTML value for "greater than" sign (>).

##HTML_AMP

Constant HTML value for "ampersand" sign (&).

##HTML_QUOT

Constant HTML value for "double quote" sign (").

##HTML_REG

Constant HTML value for "Registered Trademark" sign.

##HTML_COPY

Constant HTML value for "copyright" sign.

##HTML_NBSP

Constant HTML value for "no page breaking" space (' ').

Variables Defined by Value

All input variables are defined BY VALUE, that is, every value which is MOVE compatible can be used, especially strings.

Creating a Next Page

If your output possibly exceeds the limits of your return page, use the subroutine W3COUNTER in the library SYSWEB to evaluate how many bytes are free in the return page.

Changing the Amount of Data Transferred

To change the amount of data transferred between browser and Natural, go through the following steps:

1. Set the following variables and parameter to the same value:
 - the upper bound of the variable `##HTTP_RETURN_PAGE` in the global data area `W3CONTEX`;
 - the upper bound of the variable `RETURN_PAGE` in the parameter data area `W3PARM`;
 - parameter `RPC_INOUT_LENGTH` in the `.ini` file of the used Natural Web Server Extension program.
This defines the maximum length of a generated page.
2. Set the following variables to the same value:
 - the upper bound of the variable `##HTTP_ENVIRONMENT` in the global data area `W3CONTEX`;
 - the upper bound of the variable `ENVIRONMENT` in the parameter data area `W3PARM`.
This defines the maximum length of the data received. This value must be smaller than or equal to the maximum length of a generated page (defined in Step 1).
3. Recatalog `W3CONTEX`, `W3PARM` and `W3ACCES` in library `SYSWEB` (`W3ACCES` encapsulates all calls to `W3CONTEX`).
4. Recatalog all subprograms which are to be called via the Natural Web Server Extension and all programs called `NAT-*`, `E3*`, `D3*`, `D4*` and `Web*` in the library `SYSWEB`.

Testing Subprograms

There are three ways to test your subprograms:

1. Call the subprogram from your web browser.
2. Call the subprogram `NAT-DIR` in the library `SYSWEB` to see the contents of a Natural library. You can also specify the name of the library in the parameters, for example `http://.../sysweb/NAT-DIR?LIB=SYSEXT`. Click on your program to start it.
3. If you do not want to call your subprogram from the web, you can use the Natural program `WEB-ONL` to simulate a remote call. The output of this program will be saved as a Natural text object. This "online execution" allows you to use the Natural Debugger.

The Natural Web Server Extensions

The Natural Web Server Extension is called from a HTTP server. The program repackages the parameters it receives from the HTTP server and performs an Entire Broker RPC or a DCOM call to the specified Natural subprogram or method.

Parameters

Data sent by the HTTP server is recognized and preprocessed. The URL, which was transmitted to the HTTP server in a URL-decoded (modified) form, is reset to its original state. All non-binary data can be transmitted as data and will be converted from ASCII to EBCDIC and vice versa, if necessary.

Initialization File

Only variables specified in your HTML page will automatically be transferred to the subprogram called. All other variables to be transferred must be specified in an ENV= entry of the .ini file. In this way, it is possible to add variables which will be treated as system environment variables. To add a system environment variable, specify a SETENV= entry in the .ini file.

Example .ini file

```
ENV=HTTP_REFERRER
ENV=HTTP_HOST
;
SETENV=VERSION:=alpha
SETENV=BROKER:=local
```

Error Logging

To save the last HTML page that was transmitted from the server to a file, specify the TRACE_FILE parameter in your configuration file.

To return an error log, specify the ERROR_LOG_FILE parameter as log-file name in your configuration file.

To get your own error screen, specify the ERROR_TEMPLATE parameter in your configuration file with your desired HTML error page's name. Environment variables can be specified in the HTML error page by using the prefix "\$". With the environment variable \$NWW_ENVIRONMENT, all environment variables transmitted to the subroutine called will be written as comment lines to the error page.

Data Areas

The subprograms in the Natural server environment have to use the parameter data area W3PARAM from the library SYSWEB.

```
DEFINE DATA
PARAMETER USING W3PARAM
...
END-DEFINE
```

The size of two specific variables in the global data area W3CONTEX has to be equal to the size of two specific variables in the parameter data area W3PARAM:

GDA W3CONTEX	PDA W3PARAM
##HTTP_RETURN_PAGE(A250/1:a)	##RPC.RETURN_PAGE(A250/1:a)
##HTTP_ENVIRONMENT(A250/1:b)	##RPC.ENVIRONMENT(A250/1:b)

The size of the variable `##RPC_RETURN_PAGE` in the parameter data area `W3PARM` has to be equal to the value specified in `RPC_INOUT_LENGTH` defined in the configuration file of the calling program.

Naming Conventions of the Library SYSWEB

Subroutines W3*

W3* subroutines access the interface to your HTTP server in the Natural Web Server Extension. Such an interface consists (basically) of a parameter data area and of a log of the data transmitted. The W3* subroutines used by the subprogram are called by the HTTP server using the Natural RPC.

Subroutines H3*

If you call one of the H3* subroutines from one of your subroutines, it creates a basic HTML tag.

Subprograms NAT*

There are some utilities to be called from the Internet.

Natural Text Members T3*

They describe what the Library SYSWEB contains, what the subroutine names are and which parameters can be passed. They also provide a code sample of how to invoke them. Use the utility `nat-docu` to access this online documentation.

Subprograms E3*

Sample code of the online documentation.

Members D3* and D4*

They are two demonstration applications.

Programs Web*

They are utilities that run from the Natural NEXT prompt.

Conversion Program: HTML to Natural

Not applicable to mainframe systems.

This section describes the use of HTML to Natural, a program that enables you to convert an HTML page into a Natural subprogram for use with Natural Web Interface.

Using HTML to Natural to generate Natural code from an HTML page avoids you having to adapt HTML input to the conventions of Natural code. You can then move the "HTML-page-turned-subprogram" to the server, including all the other Natural program logic you have added. If you want to change the HTML page again, go back to your source, convert it and move it to the server again. This is much easier than writing HTML with a browser, moving it to the server, adding Natural program logic and reiterating the process if your HTML page changes.

This section covers the following topics:

- Using the Conversion Program
- Inserting a Natural Tag
- Options
- Generating a DCOM Class
- Online Test Utility WEB-ONL

Using the Conversion Program

If your basic Web pages are designed with editing tools, it takes some effort to include such a page in a Natural subprogram that can be called from the Web.

"HTML to Natural" is a program that uses an HTML page as input and generates a Natural subprogram, which can be called by the Natural Web Server Extensions using the Natural Web Interface.

```

09:57:28          ***** HTML to Natural *****          2002-01-14
                   - Main Menu -                          Library SYSWEB

Input File:
/nat/natc/511/samples/sysweb/*.htm_____

Output to Natural
Library ..... SYSWEB
Object type ..... N
Object ..... _____
Subroutine name .. _____

Select HTML file for generation.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit      Class      Test Opt. Save Canc
  
```

Below is information on:

- Functions and PF Keys
- Generating a Subprogram/Subroutine to be called direct from the Web

Functions and PF Keys

PF Key	Function	Explanation
PF1	Help	Invokes the Help function for the field at which the cursor is positioned.
PF3	Exit	Leaves the program and returns to the command line.
PF6	Class	Starts the program that generates a DCOM Class (see the relevant section).
PF9	Test	Starts the Online Test Utility (see the relevant section).
PF10	Opt.	Options. Specifies options for the generation process.
PF11	Save	Saves the selected input and output files as default Natural parameters.
	Next	Starts generating the program.
ENTER		

Generating a Subprogram/Subroutine to be called direct from the Web

 To generate a subprogram/subroutine to be called direct from the Web

1. Select your HTML page.
2. Close the Natural library you want to generate.
3. Select the object type you want to generate.
4. Select your Natural file name.
5. Start the generation.
6. After generation, you can call the Natural Web Interface to show the page.

Inserting a Natural Tag

If you use Natural on your HTML page, it is possible to specify your special Natural coding direct in the HTML page. After generation, the program needs no additional changes.

The HTML2NAT program can recognize a <NATURAL> tag. All lines between <NATURAL> and </NATURAL> will be copied, as they are, to the generated Natural source object.

Appearance

<NATURAL> </NATURAL>

Below is information on:

- Attributes DATA, LDA, GDA, SUB, NOT
- Comment Tag
- ASP-like Script Commands
- Additional Script Directives
- Example 1 of a Simple Generation
- Example 2 of a Simple Generation with a Natural Tag

Attributes DATA, LDA, GDA, SUB, NOT

Listed below are attributes provided to define coding sections that are to be moved within the program or excluded from the program.

Attribute	Explanation
DATA	<NATURAL DATA> or <NATURAL LDA> moves the defined section to the DEFINE DATA LOCAL part of your program.
LDA	
GDA	<NATURAL GDA> moves the defined section to the DEFINE DATA GLOBAL part of your program.
SUB	<NATURAL SUB> moves the defined section to the end of the program. This enables you to specify inline subroutines.
NOT	<NATURAL NOT> excludes the defined section from the program. This enables you to specify the design of part of a page that will be generated by a program.

Comment Tag

Use the comment tag "<!-- -->" to hide the display of defined sections of your coding. If you use the comment tag and <NATURAL NOT>, you can display the predefined page with a normal browser. This helps you to specify your page and replace parts of the page dynamically.

ASP-like Script Commands

With the new Natural Version 5.1 of HTML2NAT, not only <NATURAL> and </NATURAL> can be used but also ASP-like (Active Server Page) script commands which are differentiated from the text by using the "<%" and "%>" delimiters.

Additional Script Directives

The following Natural-specific directives must be used when writing a Natural subprogram:

Output directive: <%= ... %>
 Short form for <% PERFORM W3HTML ... %> tag

Subprogram directive: <%SUB ... %>
 equal to the <NATURAL SUB> ... </NATURAL> tag

Global Data Area directive: <%GDA ... %>
 equal to the <NATURAL GDA> ... </NATURAL> tag

directive: <%LDA ... %>
 equal to the <NATURAL LDA> ... </NATURAL> tag

Not directive: <%NOT ... %>
 equal to the <NATURAL NOT> ... </NATURAL> tag

Processing directive <%@ LANGUAGE=NATURAL %>
 indicates that the used language is Natural..

Example 1 of a Simple Generation

HTML document:

```

<HTML><HEAD><TITLE>
Example1 genNat
</TITLE></HEAD><BODY><H2>
Example1 genNat
</H2><HR>
<P>This is for your output
</BODY></HTML>

```

Generated Natural subprogram:

```

* ----- SUBPROGRAM generated out of file:
* ----- C:\example1.html
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
LOCAL
* ----- PRIVATE VARIABLES -----
1 W3VALUE (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
PERFORM W3ERROR ##W3ERROR
PERFORM W3END ##RPC
ESCAPE ROUTINE
END-ERROR
* ----- INITIALIZE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
* ----- MAIN PROGRAM -----
PERFORM W3TEXTLINE '<HTML><HEAD><TITLE>'
PERFORM W3TEXTLINE 'Example genNat'
PERFORM W3TEXTLINE '</TITLE></HEAD><BODY><H2>'
PERFORM W3TEXTLINE 'Example genNat'
PERFORM W3TEXTLINE '</H2><HR>'
PERFORM W3TEXTLINE '<P>This is for your output'
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
*
* ----- SUBROUTINES -----
END

```

Example 2 of a Simple Generation with a Natural Tag

HTML document:

```
<HTML><HEAD><TITLE>
Example2 genNat
</TITLE></HEAD><BODY><H2>
Example2 genNat
</H2><HR>
<P>This is for your output
<HR>
<P>generated at:
<NATURAL NOT>
Time/Date
</NATURAL>
<NATURAL><!--
  PERFORM DOTIME
--></NATURAL>
<NATURAL SUB><!--
DEFINE SUBROUTINE DOTIME
  COMPRESS *TIME *DATE INTO #VALUE
  PERFORM W3TEXTLINE #VALUE
END-SUBROUTINE
--></NATURAL>
<NATURAL DATA><!--
1 #VALUE (A30)
--></NATURAL>
</BODY></HTML>
```

Generated Natural subprogram:

```

* ----- SUBPROGRAM generated out of file:
* ----- C:\example2.html
DEFINE DATA
PARAMETER USING W3PARAM
LOCAL USING W3CONST
1 #VALUE (A30)
* ----- PRIVATE VARIABLES -----
1 W3VALUE (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALIZE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
* ----- MAIN PROGRAM -----
PERFORM W3TEXTLINE '<HTML><HEAD><TITLE>'
PERFORM W3TEXTLINE 'Example2 genNat'
PERFORM W3TEXTLINE '</TITLE></HEAD><BODY><H2>'
PERFORM W3TEXTLINE 'Example2 genNat'
PERFORM W3TEXTLINE '</H2><HR>'
PERFORM W3TEXTLINE '<P>This is for your output'
PERFORM W3TEXTLINE '<HR>'
PERFORM W3TEXTLINE '<P>generated at:'
  PERFORM DOTIME
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
*
* ----- SUBROUTINES -----
DEFINE SUBROUTINE DOTIME
  COMPRESS *TIME *DATE INTO #VALUE
  PERFORM W3TEXTLINE #VALUE
END-SUBROUTINE
END

```

Note:

The syntax of the Natural program will not be checked during conversion.

Options

```
14:04:47          ***** HTML to Natural *****          2002-01-14
User SAG              - Options -                          Library SYSWEB

HTML File
Delete unnecessary white space ..... _
Save .. at source ..... _

Generated Source
Stow after generation ..... _
Natural line length ..... 128

Default input file:
$NATDIR/$NATVER/SAMPLES/SYSWEB/*.HTM_____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help           Exit                                           Canc
```

Below is information on:

- Input/Output Fields
- Functions and PF Keys

Input/Output Fields

Field	Explanation
Delete unnecessary white space	If checked, multiple white-space characters such as blank, new line, tab, will be reduced to a single white space. For special HTML tags such as <PRE> <TEXTAREA> or <SCRIPT>, the white space will not be collapsed. Default value: unchecked
Save <NATURAL NOT> ... <NATURAL> in Source	If checked, the content of <NATURAL NOT> tags will not usually be generated into the Natural source. This option generates the content of <NATURAL NOT> as comment into the Natural source. Default value: unchecked
Stow after Generation	If checked, the generated program will be stowed if the generation has been successful. Default value: checked
Natural Line Length	The length of the generated Natural source lines: the minimum value is 20, the maximum 248. Default value: 72
Default Input File	The default input file to be used for the generation. Default value: /nat/natc/511/samples/sysexxt

Functions and PF Keys

PF Key	Function	Explanation
ENTER		Leaves the program and saves the changes.
PF3	Exit	Returns to the command line.
PF12	Canc	Leaves the program without saving your changes.

Generating a DCOM Class

If the Natural Web Interface subprograms should be called using DCOM instead of RPC, a DCOM class is needed. This class contains as methods all relevant Natural subprograms for the Natural Web Interface.

The program HTML to Natural automatically generates the specified class. To stow the generated class, a Local Data Area (LDA) is needed to specify the Global Unique IDs (GUIDs) of the DCOM objects. The name of the LDA begins with **L** which is followed by the first seven characters of Library Name.

Below is information on:

- Invoking Generate Class
- Input/Output Fields
- Example for Library SYSWEB
- Functions and PF Keys

Invoking Generate Class

 **To invoke the screen Generate Class for Web Interface**

1. Start program WEB-R2DC
Or open program HTML2NAT.
2. Press PF6.

```

09:44:02          ***** Generate Class for Web Interface *****          2002-01-14
                                - Main Menu -                                Library SYSWEB

Library ..... SYSWEB__
Class object .... SYSWEB__
Class name ..... SYSWEB_____
LDA name ..... LSYSWEB_

Class found - Press Enter to start generation.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help           Exit                                           Canc
    
```

Input/Output Fields

Field	Explanation
Library	The name of the library to be scanned.
Class Object	The name of the class source. We recommend that the name you choose for Class Source is identical to the name of the library.
Class Name	The name of the class that can be called later from the Internet. We recommend that the name you choose for Class Name is identical to the name of the library for which the class is generated.
LDA Name	The name of the LDA containing the GUIDs for the class ID and the Natural Web Interface ID. For the naming conventions that apply, see Example for Library SYSWEB below.

Example for Library SYSWEB

The LDA name is LSYSWEB. Name the first GUID CLSID- followed by the Library name and the second GUID IID-NATWEB.

	T		Comment			
			*** Top of Data Area ***			
X	U	1	CLSID-SYSWEB	A	36	
X	U	1	IID-NATWEB	A	36	
			*** End of Data Area ***			

Attention:

Do **not** copy and rename or move an LDA in order to get new GUIDs for your classes. If an LDA is copied and renamed or moved, the preset GUID is not changed. This may cause major problems.

Functions and PF Keys

PF Key	Function	Explanation
ENTER		Generates the class and leaves the program.
PF3	Exit	Returns to the command line.
PF4	Defau	Sets default values. This function is enabled if no relevant class is found for the library. The defaults for class source and class name are given. The LDA needed has to be generated in advance.
PF12	Canc	Leaves the program without generation.

Online Test Utility WEB-ONL

This Test Utility is a component of Natural Web Interface. You have the ability to check your subprogram locally without involving an HTTP server. The transfer parameters for your Web page are transferred into the Test Utility and are posted directly to the business logic. As communication platform, you can choose either RPC or DCOM as in real remote communications. The result is either the Web page expected or an error message. The Web page can be viewed with the browser or a viewer of your choice. If you receive an error message, you can easily debug your business logic locally without writing an extra test routine. No remote debugging is needed.

Features:

- Local application check.
- No need for remote debugging.
- Simplified error checking.
- No need to write an extra test routine.

Below is information on:

- Running the Application
- Input/Output Fields
- Functions and PF Keys

Running the Application**▶ To run the application**

1. Start the program WEB-ONL.
2. Select a library and subprogram name.
3. Optional: add parameters.
4. Choose RPC or DCOM.
5. Press ENTER.

```

09:55:24          ***** Natural Web Online Test Utility *****          2002-01-14
                    - Main Menu -                                     Library SYSWEB

Library ..... SYSWEB_
Subprogram ..... NAT-ENV
DCOM/RPC ..... R
Output text object .. WEB-OUT_

Parameter
  Name                               Value
1: _____
2: _____
3: _____
4: _____
5: _____
6: _____
7: _____
8: _____
9: _____

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help           Exit                                     Canc
    
```

Input/Output Fields

Field	Explanation
Library	The library in which the required subprogram is stored.
Subprogram	The name of the required subprogram.
DCOM/RPC	Can be selected with either DCOM or RPC as communication form. For DCOM, you have to register your classes first. Default: R
Output Text Object	The name of the Natural object of the type Text that stores the result of the generated Web page. Default: WEB-OUT
Parameter: Name Value	Here you can enter the name-value-pairs needed from the subprogram. If you use server parameters, place an ampersand (&) in front of the variable name before you add the parameter to the parameter list.

Functions and PF Keys

PF Key	Function	Explanation
ENTER		Runs the process of receiving the output of the requested subprogram. The status of the process can be seen in the message line at the bottom of the WEB-ONL program screen.
PF3	Exit	Leaves the Test Utility and returns to the command line.
PF12	Canc	Cancel. Stops processing.

Programming Tips

This section provides some tips on using the Natural Web Interface.

This section covers the following topics:

- Editing in Lower Case
 - Quote v. Apostrophe
 - Variables defined by Value
 - Access to Resources
 - Constant Values
 - Creating a New Page
 - DCOM / RPC
-

Editing in Lower Case

If you use Natural on a mainframe, you may set at your Editor the following:

Set your Editor in Lower Case

1. Follow the following menu structure:
Profile > Additional Options > General Defaults > Editing in Lower Case
 2. Enter **Y** in the field **Editing in Lower Case**.
- All programs delivered with the Natural Natural Web Server Extension use ' (quotation) and " (double quotation) in a way, that conversion to uppercase depends on which pair of characters is used.
 - Strings surrounded by pairs of ' (quotation) will not be converted to upper case and strings surrounded by pairs of " (double quotation) will be converted.

Quote v. Apostrophe

To use both quote and apostrophe within your application, check the Natural parameter Translation of quotations marks (TQ). This parameter controls the translation of a quotation mark (") within a Natural text constant. It takes effect at compilation time only. Turn this parameter to OFF or use W3-QUOTE-DQUOTE.

Parameters

```
1 W3QUOTE           (A001) /* o/ : Quote (")
1 W3APOSTROPHE     (A001) /* o/ : Apostrophe (')
```

How To Invoke

```
PERFORM W3-QUOTE-DQUOTE W3QUOTE W3APOSTROPHE
```

Variables defined by Value

All input variables are defined **BY VALUE**, this means, every value which is **MOVE** compatible can be used, especially constant strings.

Access to Resources

All resources, such as pictures, sounds or Java applets, are saved at the HTTP server. If you want to create and relocate the program, do not hardcode the pathname of these resources.

When defining an environment variable, you specify the current path of the resource. The environment variable can be set at the Natural Web Server Extensions. If no variable is set, use a default setting.

Constant Values

The parameter data area W3CONST contains some useful constant values:

##HTTP_END

All routines which write their output to the return page remove trailing blanks. If no removing should be done, compress a ##HTTP_END string at the end of your string. The output processing will increase.

##HTTP_NEWLINE and ##HTTP_NEWLINE_END

Writing to the return page, a physical new line can be created by compressing the string ##HTTP_NEWLINE into the string. To avoid stripping, you can also use ##HTTP_NEWLINE_END.

##HTTP_NEWLINE_LENGTH

The length of the string ##HTTP_NEWLINE may differ for different implementations. Use ##HTTP_NEWLINE_LENGTH if the length of ##HTTP_NEWLINE is needed. The length of string is ##HTTP_NEWLINE_LENGTH+1.

Creating a New Page

If your output may exceed the limits of your return page, use W3COUNTER to evaluate how many bytes are free at the return page.

DCOM / RPC

When you write an application that works with both RPC and DCOM, there are some aspects you should consider:

- Do not exceed the name sign limitation for Natural libraries and subprograms. With the DCOM interface, you can use up to 32 characters to name a class and its methods (see NaturalX documentation).
- Use the same name for a class and the library into which all your subprograms are located. This may not be according to object-oriented design principles, but gives you the possibility to access your subprograms via RPC or DCOM. EntireX supports a dynamic logon to a given Natural library.
- Now the library is the equivalent to a class, and all programs contained in that library are the methods of this class. Calling with RPC is now ready. To call with DCOM, you only have to specify all subprogram as methods of your class.
- With the Natural Web Interface, a program called W3-R2DC(SYSWEB) to generate a class for a Natural library is delivered. The program checks all subprograms if W3PARM is used as parameter data area and includes these subprograms as methods to the generated class.

Web Interface Administration

This section covers the following topics:

- Set the Size of the Return-Page Transport Buffer
 - Set the Size of the Return Page
 - Create a User-defined Error Page
 - Create a User-defined Error Page XML-Style
 - Alphanumeric-to-HTML Conversion
 - Alphanumeric-to-URL Conversion
-

Set the Size of the Return-Page Transport Buffer

Changing the Transport Send Buffer Width

To change the transport send buffer width:

1. Change the upper bound of the variable RETURN_PAGE in the parameter data area W3PARM. Use this value for the parameter NWW_INOUT_LENGTH in the initialization file used for the Natural Web Server Extension program and the initialization of the value ##HTTP_RETURN_PAGE_PART in the Local Data Area W3LIMITS.
This defines the maximum length of the transport buffer.
2. Recatalog all W3* sources from library SYSWEB.
3. Recatalog all subprograms that are to be called using the Natural Web Server Extension, all NAT-*, HTTP* and NAT-* programs from the library SYSWEB.

Changing the Received Data Buffer Width

To change the received data buffer width:

1. Change the upper bound of the variable ##HTTP_ENVIRONMENT_BLOCK (divided by 250) in the local data area W3CONTEX and of variable ENVIRONMENT in the parameter data area W3PARM to the same value.
2. Use this value to initialize ##HTTP_ENVIRONMENT_MAX in the local data area W3LIMITS.
This defines the maximum length of received data.
This value must be less than or equal to the maximum length of the transport buffer (see Step 1).
3. Recatalog all W3* sources from the library SYSWEB.
4. Recatalog all subprograms which are to be called using the Natural Web Server Extension, all NAT-*, HTTP* and NAT-* programs from library SYSWEB.

Changing Your Return Page

To change your return page:

1. Change the upper bound of ##HTTP_RETURN_BLOCK (divided by 250) and ##HTTP_RETURN_PAGE in the local data area W3CONTEX
and use this value to initialize ##HTTP_RETURN_PAGE_MAX in the local data area W3LIMITS.
2. Recatalog all W3* sources from library SYSWEB.
3. Recatalog all subprograms that are to be called using the Natural Web Server Extension, all NAT-*, HTTP* and NAT-* programs from the library SYSWEB.

Set the Size of the Return Page

To change the amount of the data transferred between the HTTP server and Natural:

1. Change the upper bound of the variable `##HTTP_RETURN_PAGE` in the **global data area** W3GLOB, of variable `RETURN_PAGE` in the **parameter data area** W3PARAM and of parameter `RPC_INOUT_LENGTH` in the initialization file of the **Natural Web Server Extension program** used to the same value. This defines the maximum length of a generated page.
2. Change the upper bound of the variable `##HTTP_ENVIRONMENT` in the **global data area** W3GLOB and of the variable `ENVIRONMENT` in the **parameter data area** W3PARAM to the same value. This defines the maximum length of data received. This value must be less or equal than the maximum length of a generated page (see step 1).
3. Recatalog W3GLOB, W3PARAM and W3ACCESS from the library SYSWEB. (W3ACCESS encapsulates all calls to the W3GLOB).
4. Recatalog all subprograms which are to be called via the Natural Web Server Extension, all NAT-* and HTTP* programs from the library SYSWEB and all NAT-* and E3* subprograms from library SYSWEB.

Create a User-Defined Error Page

If a Natural error occurs and the default ON ERROR block is specified, W3ERROR will be called and a predefined error page will be generated.

If you want to change this error page, change the Subroutine W3ERROR-TEMPLATE (SYSWEB/W3ERRTMP).

This program generates a complete HTML page.

Create a User-Defined Error Page XML-Style

If a Natural error occurs and the default ON ERROR block is specified, W3ERROR will be called and a predefined error page will be generated.

If you want to change this error page to an XML-conform HTML, proceed as follows:

1. Uncatalog the subroutine (SYSWEB/W3ERRTMP).
2. Open the subroutine SYSWEB/W3ERXTMP).
3. Rename W3ERROR-TEMPLATE-XML to W3ERROR-TEMPLATE.
4. Stow the program.

This program now generates a complete XML-conform HTML page.

Alphanumeric-to-HTML Conversion

For a conversion to HTML, special characters have to be replaced by the correct HTML representation.

- The subroutine W3-ASCII-HTML-TABLE (SYSWEBP/W3AS2HT) contains the settings for the replacement of characters.
- W3INIT and H3-TEXT-TO-HTML will call W3-ASCII-HTML-TABLE.

The generated table is saved in the global data area W3GLOB for faster access. It is possible to save up to 128 replacements.

If HEX values are used for the definition (e.g. quote), a value for the ASCII and one for the EBCDIC character set has to be defined. Otherwise the file is not portable.

Alphanumeric-to-URL Conversion

For URL decoding, some special characters have to be replaced by the correct URL-conform representations.

- The subroutine H3-ASCII-URL-TABLE(SYSWEB/H3AS3URL) contains the settings for the replacement of characters.
- H3-ASCII-URL-TABLE will be called by H3-TEXT-TO-URL.

It is possible to save up to 128 replacements.

If HEX values are used for the definition (e.g. quote), a value for the ASCII and one for the EBCDIC character set has to be defined. Otherwise the file is not portable.

Demonstration Application - without JavaScript

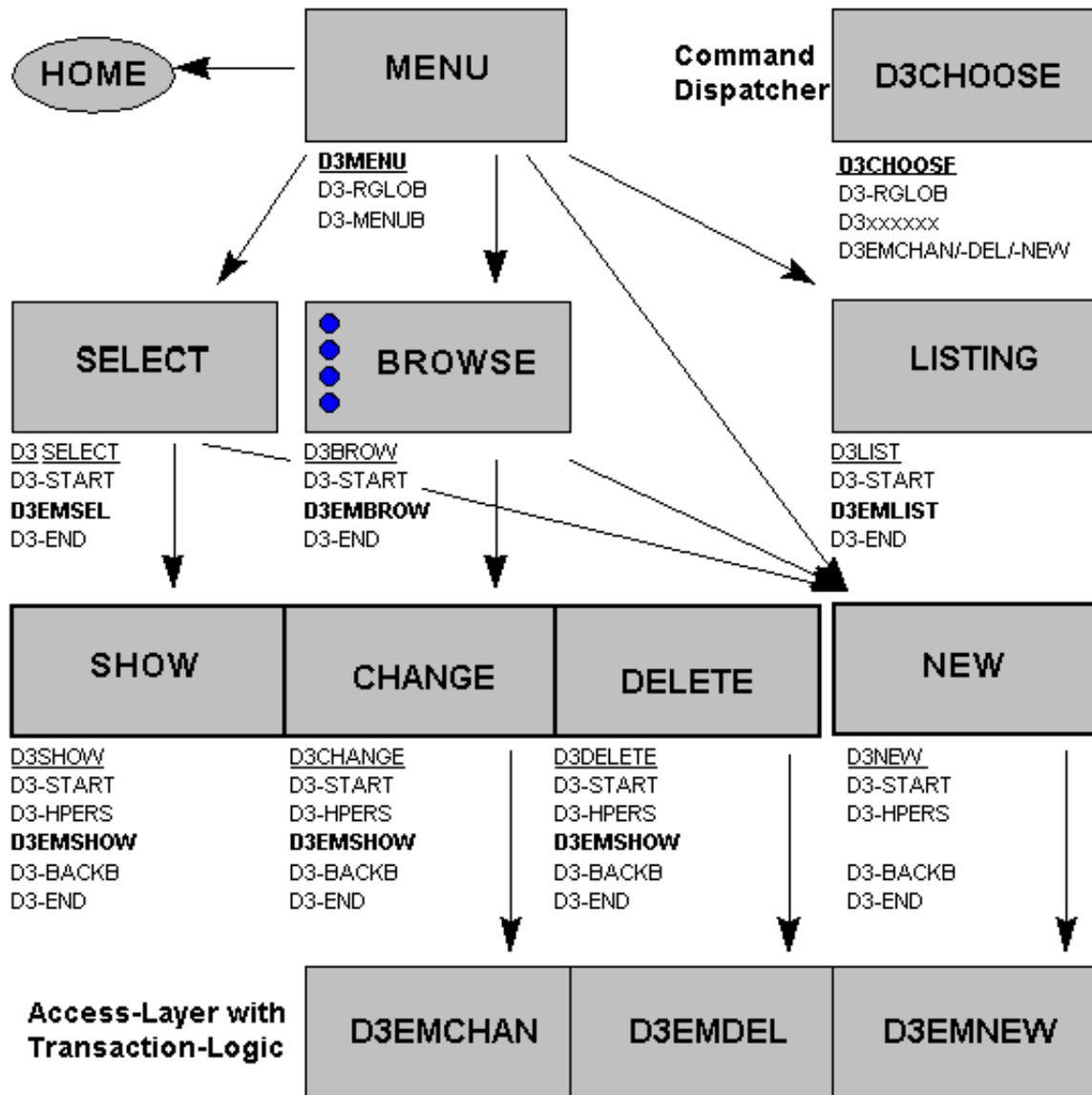
This section covers the following topics:

- Business Requirements
 - Design Decisions
 - Libraries, Modules and Naming Conventions
 - Starting the Demonstration Application
 - Starting the Natural Web Interface Online Manual
 - Requirements
-

Business Requirements

The demonstration application shows the use and programming of the Natural Web Interface. The functionality includes simple file maintenance with various selection functions as shown in the graphic below.

The demonstration is platform independent and is based on the Adabas files EMPLOYEES and VEHICLES.



Legend: Module HTML-Form

Module: NATURAL object type subprogram
 → Call of dispatcher module D3CHOOSE

Design Decisions

The HTML-GUI has some restrictions for application design:

- a unique layout is not possible for different browsers
- the HTML-GUI elements have restricted functionality. For example, no input in selection box, only predefined fonts or buttons for submit (no default button)

So in the demonstration application we use:

- forms with submit buttons
- global data exchange with hidden fields on the forms
- usage of the form send back method GET (URL plus visible parameters for bookmarks)
- no usage of VB / JAVA Scripts for implementation of processing rules
- a command dispatcher module (D3CHOOSE) for navigation
- standard pictures for group/male/female persons because of copyright reasons

Libraries, Modules and Naming Conventions

The demonstration contains one module (see also the installation of the Natural Web Server Extension):

SYSWEB

This library contains the following modules:

- T3 HTML text for online documentation
- E3 Examples for online documentation
- D3 Demonstration application modules

Starting the Demonstration Application

The start module for the demonstration is D3MENU.

To start the demonstration application (depending on your installation of the Natural Web Server Extension), call the subprogram D3MENU in library SYSWEB.

Example of the URL to call the demonstration application:
`http://yourserver/yourcgi/sysweb/d3menu`

Starting the Natural Web Interface Online Manual

You can start the online documentation from the Natural Web Interface.

The start module for the demonstration is D3MENU.

To start the online manual, call the subprogram D3MENU in library SYSWEB.

Example of the URL to call the demonstration application:

`http://yourserver/yourcgi/sysweb/d3menu`

Requirements

The following software must be installed:

- Natural Web Server Extensions, a part of Natural Web Interface.
- Adabas with the file EMPLOYEEES.

Perform a CATALALL for the programs D3* in the library SYSWEB to activate the demonstration application.

To view the pictures of the example delivered with the Natural Web Server Extension, copy all pictures to a directory /pictures of your HTTP server

or set the environment variable PICTURES for the Natural Web Server Extension to the specific directory.

Demonstration Application - with JavaScript

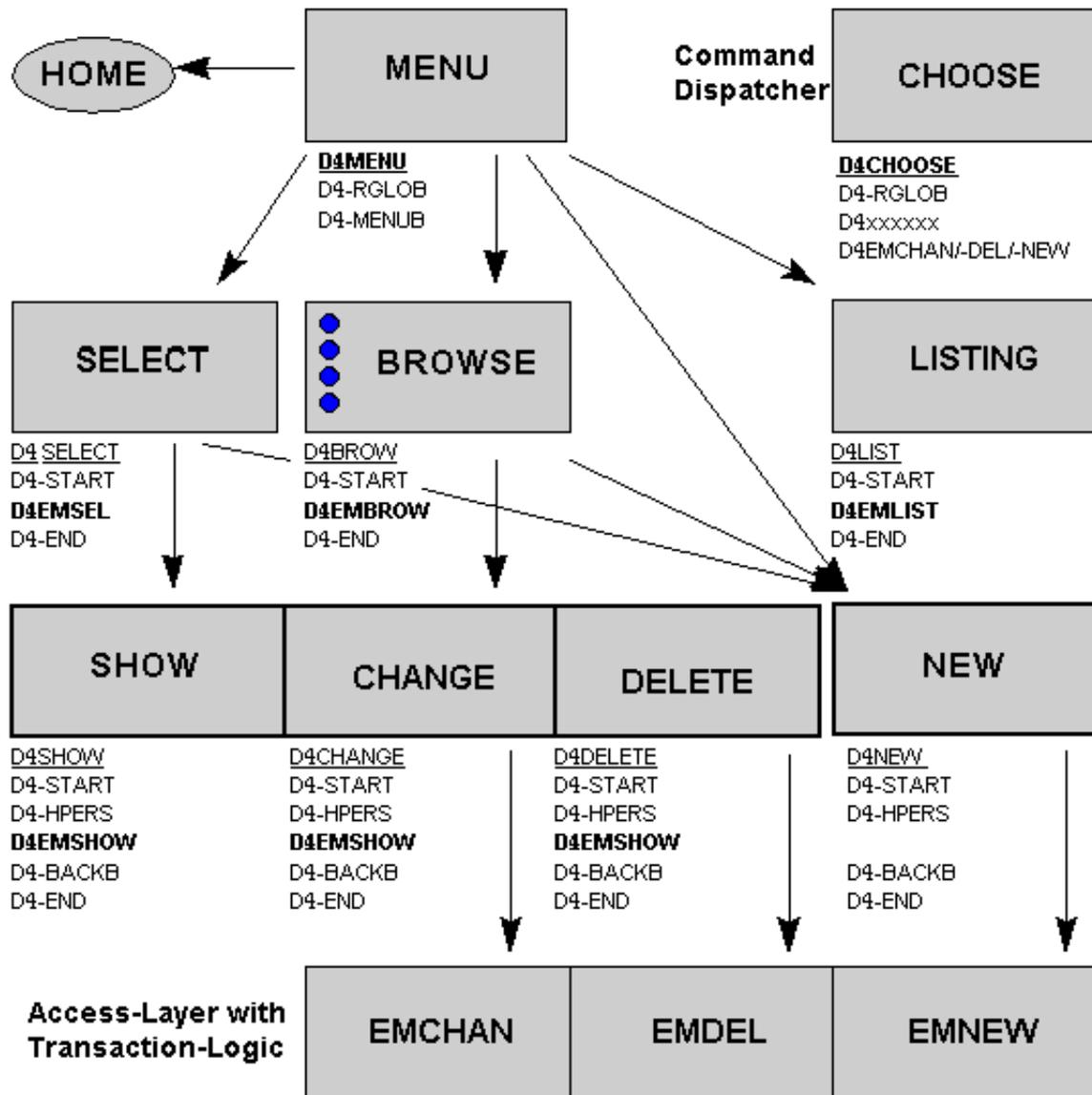
This section covers the following topics:

- Business Requirements
 - Design Decisions
 - Starting the Demonstration Application
 - Requirements
-

Business Requirements

The demonstration application shows the usage and programming of the Natural Web Interface. The functionality includes simple file maintenance with various selection functions as shown in the graphic below.

For the purpose of cross-platform availability, this demonstration is based on the Adabas files EMPLOYEES and VEHICLES.



Legend: Module HTML-Form

Module: NATURAL object type subprogram
 → Call of dispatcher module D4CHOOSE

Design Decisions

Use state of the art web design:

- Javascript
- 'global' data exchange with hidden fields on the forms
- usage of the form send back method GET (URL plus visible parameters for bookmarks)
- a command dispatcher module (D4CHOOSE) for navigation

Starting the Demonstration Application

The start module for the demonstration is D4ENTER. Depending on your installation of the Natural Web Server Extension, call the subprogram D4ENTER in library SYSWEB.

Example for the URL to call the demonstration application:

`http://yourserver/yourcgi`

Requirements

Natural Web Server Extensions, a part of Natural Web Interface, and Adabas with file Employee have to be installed. Perform a CATALALL for the programs D4* in the library SYSWEB to activate the demonstration application.

To view the pictures in the example, you must install the Natural Web Server Extension demonstration part in your HTTP Server root.

Natural Web Interface Error Messages

This section lists error messages you may receive when you are working with the Natural Web Interface. A description of each error and a solution is provided.

Error Messages

Error Number	Error Message	Description	Action
NWW9002	No elements defined.	The number of array values is set to 0.	Correct the program.
NWW9003	Can only be used inside a FORM tag.	This tag can only be used inside a FORM tag.	Initialize a FORM with H3-OPEN-FORM.
NWW9004	A FORM tag without ACTION is not allowed.	For each FORM, an ACTION has to be specified.	Correct the program.
NWW9005	LI tag outside a list not allowed.	LI has to be placed inside a list.	Initialize a FORM with H3-OPEN-LIST.
NWW9006	List nested too deep: ...	Only 10 level are supported for lists.	Decrease your level.
NWW9007	Radio Button Group has no name.	To generate a Radio Button Group, a name is needed.	Add a name.
NWW9008	Element ... has no name.	Each element of a Checkbox Group needs a name.	Add name.
NWW9009	Textarea has no name.	To generate a Textarea, a name is needed.	Add name.

Natural Web Interface Installation

The Natural Web Interface is installed in the course of the Natural for Windows NT installation procedure.

Note:

During the default installation of Natural only the Natural Web Server Extensions for cgi interfaces will be installed.

If isapi or nsapi Natural Web Server Extensions are needed, please select Custom Installation and specify the Natural Web Server Extensions you prefer.

This document contains the following sections:

-  **Configuring the Natural Web Interface** Describes how to configure the Natural Web Interface. If you are not familiar with a specific product, please read the corresponding installation instructions for more information.
-  **Troubleshooting** Provides hints for known problems.

Configuring the Natural Web Interface

This section provides information needed to configure the Natural Web Interface. If you are not familiar with a specific product, refer to the corresponding product documentation for more information.

This section covers the following topics:

- Supported HTTP Servers
- Configuring RPC and RPC Server
- Configuring the DCOM Server
- Configuring the Web Interface
- Configuring an HTTP Server
- Communication with Natural Security
- Troubleshooting

The latest documentation updates are published in ServLine24: <http://servline24.softwareag.com>. Click to [Product Documentation > Recently Added Documentation](#) and select the item from the selection box.

Supported HTTP Servers

Operating System	HTTP Server
Windows NT (Intel)	<ul style="list-style-type: none"> ● Microsoft Internet Information Server Version 4.0 ● Netscape FastTrack Server Version 3.0 ● Apache Version 1.3
UNIX (*)	<ul style="list-style-type: none"> ● Netscape FastTrack Server Version 3.0 ● Apache Version 1.3
OS/390 Unix System Services	<ul style="list-style-type: none"> ● IBM Websphere Application Server for OS/390 V2R8.0

Configuring RPC and RPC Server

In the following configuration description, ETB255 is the name of a Broker and NATWEB1 the name of an RPC Server used for the examples.

For the installation and configuration, refer to the Natural RPC, Entire Net-Work, and Entire Broker documentation.

The following topics are documented below:

- Natural Version 3.1.5 for Mainframes / Natural Version 4.1.2 for UNIX Server / Natural Version 5.1.1 for Windows
- EntireX / Entire Broker SDK

Natural Version 3.1.5 for Mainframes / Natural Version 4.1.2 for UNIX Server / Natural Version 5.1.1 for Windows

On Windows NT/2000 and UNIX Systems

To change your NATPARM file so that two additional steplibs can be accessed in the RPC environment:

- In the section **Environment Assignments**, add the two steplibs SYSWEB and SYSEXT to the steplib parameter subsection.

In an OS/390 Unix System Services Environment

If Natural Security is installed:

- Define the steplibs SYSWEB and SYSEXT for your library.

If Natural Security is **not** installed:

- Modify the Natural program WEB-STLB in library SYSWEB by entering the DBID and file number of the associated FNAT system file of the libraries SYSWEB and SYSEXT. In case of need, you can add additional steplibs.
- STOW the program.
- The STACK parameter for your RPC server should have the following value:
STACK=(LOGON SYSWEB;WEB-STLB)

EntireX / ENTIRE Broker SDK

On Windows NT / Windows 2000 Systems

Setting the environment variables is not required.

On UNIX (All Platforms)

All EntireX-relevant environment variables must be passed by the HTTP server.

Configuring the DCOM Server

To install and configure the DCOM server, proceed as described in the NaturalX documentation.

In the following configuration description, NATWEBEXT is the name of an external DCOM Server and NATWEB is the name of a local DCOM Server.

This section covers the following topics:

- DCOM
- NaturalX 3.1 Server

DCOM

On UNIX (All Platforms)

All EntireX-relevant environment variables must be passed by the HTTP server.

In an OS/390 Unix System Services Environment

The DCOM version of the Natural Web Interface Server Extension is currently not supported for OS/390 Unix System Services.

NaturalX Server

For all servers supporting the Natural Web Interface, add the libraries SYSWEB and SYSEXT as steplibs, as described above in the section Natural Version 3.1 for Mainframes / Natural Version 4.1.2 for UNIX Server / Natural Version 5.1.1 for Windows Server.

Configuring the Web Interface

Natural Web Interface

For mainframe, Windows NT, Windows 2000 and UNIX environments no configuration is needed.

Natural Web Server Extensions for RPC

Adjust the configuration file using an external editor:

```
RPC_ETB_ID_NAME=ETB255
RPC_SERVER_NAME=NATWEB1
```

In an OS/390 Unix System Services Environment

The parameter NWW_OUT_CSS_TRANSLATE must be set in the Configuration File. Its value depends on the codepage used.

Natural Web Server Extensions for DCOM

Local DCOM (All Platforms)

No adjustments are needed for local communication.

External DCOM (All Platforms)

For external communication, see the NaturalX documentation for Registry changes, or adjust the configuration file using an external editor:

```
DCOM_SERVER_NAME=NATWEBEXT
```

On Windows NT/Windows 2000 (Internet Information Server)

If you use the Internet Information Server, the username for anonymous logon, e.g. NATWEB, is used. NATWEB must belong to the group USER, or the GUEST account must be enabled.

On Windows NT/Windows 2000 (Apache)

If you use the Apache Server, the default settings for User/Group specified at httpd.conf work fine:

```
# User/Group: The name (or #number) of the user/group to run httpd as User nobody
Group #-1
```

On Windows NT/Windows 2000 (Netscape Server)

If you use the Netscape Server, for anonymous logon, the SYSTEM account is used.

To use DCOM with remote access, a specific user, e.g. NATWEB, must be used to run the HTTP server. This user must belong to the group USER and be defined on both computers.

Run Services from the Windows Control Panel to change the Logon for your HTTP Server service:

- Select Netscape Server Service > Startup...
- Log On As: yes
- Userid: NATWEB
- Password: **
- Confirm Password: **

Natural Web Server Extensions for NSAPI

Using an RPC Server

1. Install the Natural Web Server Extensions
2. Open the ...\\config\\mime.types file of the HTTP Server and add the **new line** at the end of the file:

```
type=magnus-internal/nww      exts=nww
```

3. Open the ...\\config\\obj.conf file of the HTTP Server and add the following **new lines** for the RPC Interface:

```
...
Init...
Init funcs="nww-nsapi,nww-init" fn="load-modules" shlib="nwwnsapi.dll"
Init fn="nww-init" file="<yourRoot>/nww/nsapi.ini"
...
<Object name="default">
NameTrans...
NameTrans from="/nww" fn="pfx2dir" dir=" <yourRoot>/nww" name="nww"
...
Service... method=...
Service fn="nww-nsapi" method="(GET|POST|HEAD)" type="
magnus-internal/nww"
...
</Object>...
<Object name="nww">
ObjectType fn="force-type" type="magnus-internal/nww"
Service fn="nww-nsapi"
</Object>
...
```

4. If not only one service or broker is to be used, specify other files at the /nww directory.
5. If a static read of the .ini file is wanted (performance-relevant), add the *line shown in italics* to your obj.conf.

Using a DCOM Server

1. Install the Natural Web Server Extensions.
2. Open the ...\\config\\mime.types file of the HTTP Server.
3. Add the **new line** at the end of the file:

```
type=magnus-internal/nww      exts=nww
type=magnus-internal/nwwd     exts=nwwd
```

4. Open the ...\\config\\obj.conf file of the HTTP Server.
5. Add the following **new lines** for DCOM:

```
...
Init...
Init funcs="nwwd-nsapi,nwwd-init" fn="load-modules" shlib="nwwdnsapi.dll"
Init fn="nwwd-init" file="<yourRoot>/nwwd/nsapi.ini"
```

```

...
<Object name="default">
NameTrans...
NameTrans from="/nwwd" fn="pfx2dir" dir=" <yourRoot>/nwwd" name="nwwd"
...
Service... method=...
Service fn="nwwd-nsapi" method="(GET|POST|HEAD)" type="
magnus-internal/dnww"
...
</Object>...
<Object name="nwwd">
ObjectType fn="force-type" type="magnus-internal/nwwd"
Service fn="nwwd-nsapi"
</Object>
...

```

6. If not only one service or broker is to be used, specify other files at the /nwwd directory.
7. If a static read of the .ini file is wanted (performance-relevant), add the *line shown in italics* to your obj.conf.

Configuring an HTTP Server

Windows NT/Windows 2000 (Internet Information Server 4.0)

If you use the Internet Information Server, the username for anonymous logon, e.g. |USR_NATWEB, is used.

|USR_NATWEB must belong to the group USER, or the GUEST account must be enabled.

Communication with Natural Security

The new version EntireX Broker SDK supports the usage of two passwords and userids.

The first userid is used to get access through EntireX Security and the second for Natural Security.

The HTTP Server Security is involved as a third security system.

HTTP Server Security:

Restrict the access of the NWW interface at your HTTP Server. For details, refer to your HTTP server documentation.

EntireX Security:

In the configuration File the NWW_USER_ID, NWW_PASSWORD has to be specified.

Natural Security:

A second UserId/Password (RPC_USER_ID, RPC_PASSWORD) has to be set.

If the parameter USE_REMOTE_USER is activated, the RPC_USR_ID will be set/overwritten. The RPC_PASSWORD remains unchanged.

It is necessary to setup Natural Security with "AUTO=ON" to pass security without Password. If no RPC_USER_ID/RPC_PASSWORD pair is set, the NWW_USER_ID/NWW_PASSWORD will be used to ensure the compatibility with the existing implementation.

Web Interface Troubleshooting

This section provides information on known problems:

Error	Description	Recommended Action
NWW0003 .ini File not found.	NWW initialization file not found.	Check your server extension initialization file: <ul style="list-style-type: none"> ● It has to have the same name as the executable with the extension .INI ● The server extension initialization file has to be placed at the same directory as the server extension executable. ● If server extension can be started from the DOS prompt and does not run called by the HTTP Server, then check if the .INI file can be found if it is copied to the same directory your HTTP server is started from.
NWW0011 ERX error 80010014 occurred. Severity = Error Facility = 65536 Returncode = 20 Subfacility = 3 Location = 0 Message: ERX_E_SERVICE_NOT_AVAILABLE - ETB error code 00070007	Natural RPC Server not started/found.	Check your RPC Server: <ul style="list-style-type: none"> ● Start your Natural RPC Server ● or check your RPC_SERVER_NAME at the NWW initialization file.
NWW0011 ERX error 80010014 occurred. Severity = Error Facility = 65536 Returncode = 20 Subfacility = 3 Location = 0 Message: ERX_E_SERVICE_NOT_AVAILABLE - ETB error code 02150148	Broker not started/found.	Check your Broker: <ul style="list-style-type: none"> ● Start your Broker and Natural RPC Server ● or check your RPC_SERVER_NAME and RPC_ETB_ID at the NWW initialization file.
Processing of subprogram TEST in library W3RPCDMO failed. Message: Status = O, Library = W3RPCDMO, Program = NATSRVD , Level = 01, Error = 00082, Line = 4190 Subfacility = 255 Location = 0	The program you have called does not exist or is not accessible. At the moment it is not possible to switch dynamically the Natural libraries.	Check your Natural: <ul style="list-style-type: none"> ● Does the Program really exist? ● If the program does exist, check your Logon library or the steplib or your NATPARM if the given library is included.
Natural RPC Server crash. Test with WEB-ONL on the same subprogram gets: WEB-ONL 1420 NAT0937 Conflicting array def.in parm.3 (Subprogram ..).	Natural RPC does not check the boundaries of arrays.	Recatalog your Programs.

Error	Description	Recommended Action
Demonstration application does not work.	You use different file numbers.	Recatalog the library SYSWEB.
NAT3048 File/USERID not available at open time.	Natural uses same ETID for different sessions.	Set your ETID parameter to \$. This generated a new ETID for every running Natural.

Natural Web Interface Programming Guide

This document is intended for anyone who writes applications for the Natural Web Interface.

This document provides basic information on various aspects of programming the Natural Web Interface. You should be familiar with HTML and Natural before you start to write Natural Web Interface applications.

For better understanding of the complex design of web applications, this section describes the design and implementation of the demonstration application delivered with Natural Web Interface.

This document comprises the following sections:

- Specification First we describe which parts the demonstration application should contain, and also the requirements for the running system.
 - HTML Design Not all of your ideas can be realized easily with HTML. Find out what aspects of your default appearance can be realized.
 - Program Design Now it is time to begin the real programming. But first we have to consider the overall design of our programs.
 - Implementation The specification is ready. The design is done. Let us do the work.
 - Fine Tuning What else can be done to have an application that is easy to use.
-

Example Programs

The Natural Web Interface Programming Guide contains several examples of Natural programs, as well as references to further example programs not shown in the documentation.

All these programs are also provided in source-code form in the Natural library "SYSWEB". (The programs are all written in structured mode.)

Please ask your Natural administrator about the availability of these libraries at your site.

The example programs use data from the Adabas files "EMPLOYEES" and "VEHICLES", which are supplied by Software AG for demonstration purposes.

Web Interface Specification

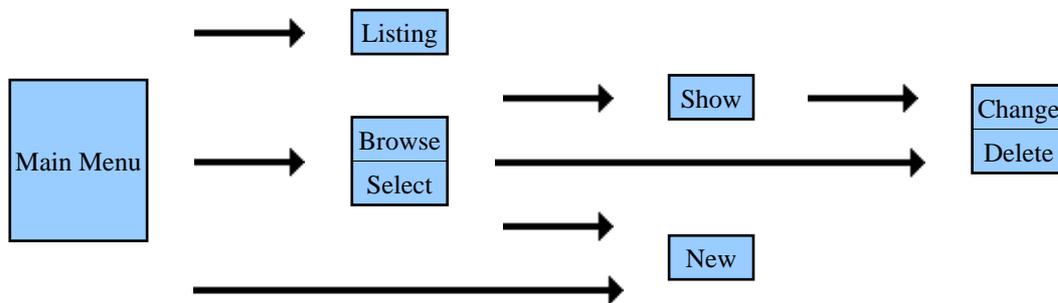
Here we describe which parts our demonstration application should contain, and requirements for the running system.

This section covers the following topics:

- Demonstration Application
- Main Menu
- Listing
- Browse
- Select
- New
- Show
- Change
- Delete
- Layout

Demonstration Application

The demonstration application should implement a simple file maintenance with selection functions and listings (see the figure below).



Adabas should be used as database, with the files "EMPLOYEES" and "VEHICLES". The "VEHICLES" example should not be implemented in the first step.

All output should be done on a standard HTML browser, capable of JavaScript, HTML 4.0 and Cascading Style Sheets.

It should be possible to change the complete environment consisting of HTTP server, Natural and communication (EntireX RPC/DCOM).

Main Menu

In the Main Menu, it should be possible to specify a file name and a range. This selection should be passed to all parts of the demonstration application

Listing

Listing should display the full record saved for an employee:

Personnel-ID First-Name Name Sex Birth Country Zip City Area-Code Phone Department

Browse

With Browse, it should be possible to see detailed information about a person.

First-Name Name Department Area-Code Phone

In addition, an employee's record should be able to be selected and then update, save, delete should be performed on this record.

Select

Select gives you the possibility to see all selected records in one list. The values displayed are:

Name First-Name

In addition, an employee's record should be able to be selected and then update, save, delete should be performed on this record.

New

With New, a record for a new employee can be added. Each record should contain the following values:

Personnel-ID First-Name Name Sex Birth Country Zip City Area-Code Phone Department

Personnel-ID is unique among all records. Each value should be checked for consistency.

Show

Show displays all values of an record:

Personnel-ID First-Name Name Sex Birth Country Zip City Area-Code Phone Department

Change

With Change, all values except the Personnel-ID can be updated.

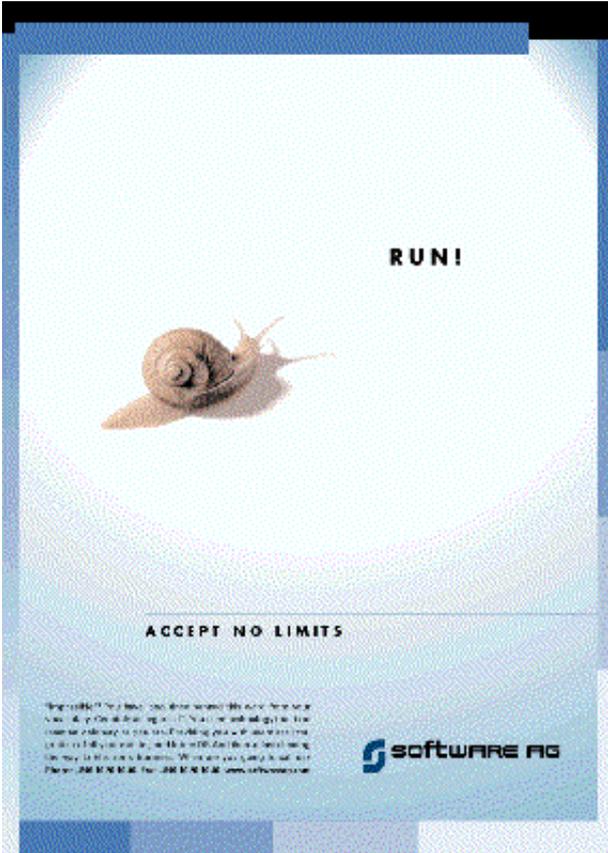
First-Name Name Sex Birth Country Zip City Area-Code Phone Department

Delete

The record will first be displayed, and after confirmation, deleted.

Layout

The application should be based on the corporate design of Software AG.



Web Interface HTML Design

This section covers the following topics:

- Given Layout as HTML Table
 - Restrictions
 - Chosen Layout
 - Input Layout
 - Functional Parts
 - How to Use Frames
 - How to Use JavaScript
-

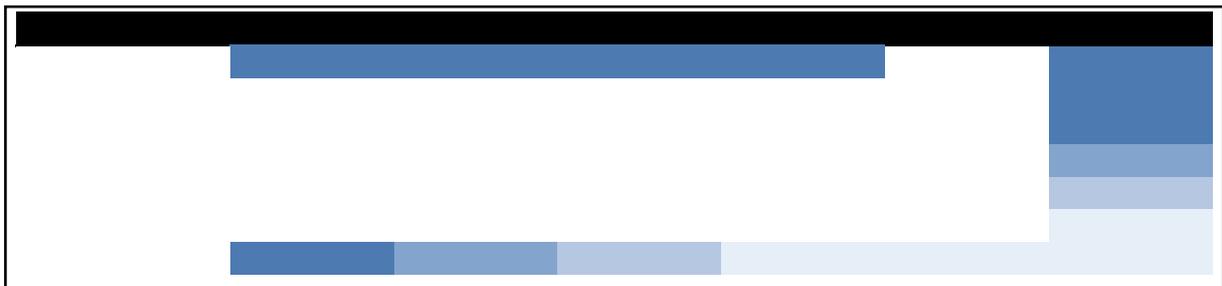
The Given Layout as HTML Table

If an HTML page is to be generated with the given layout, HTML 4.0 with extensive use of Cascading Style Sheets (CSS) is needed. In addition, the user needs a display with more the 256 colors.

If this is available, the figure below shows the result of the design. The page consists of a Table with 11 rows and 7 columns. The border on the left side is realized with one merged table field with a background picture. The picture is repeated in the x direction. All other borders are separate fields with a fixed background color. The area in the middle of the page is a merged area of 5 rows and 5 columns, filled with a background picture. The small space to the right of the headline border is filled with a small picture to give the right appearance.

The pictures used are saved in jpg format, to ensure that the correct color settings are used. For example the picture used for the middle pages used 1036 different colors.

To force the HTML browser to leave the right amount of space, a "single-pixel-GIF" is used. A transparent picture containing only one pixel is resized to the space you need.




```

    <! right border line>
    <TD bgcolor='#4D7AB1'>&nbsp;</TD>
</TR>
<!-- 8st line -->
<TR>
    <! right border line>
    <TD bgcolor='#83A4CD'>&nbsp;</TD>
</TR>
<!-- 9st line -->
<TR>
    <! right border line>
    <TD bgcolor='#B6C7E1'>&nbsp;</TD>
</TR>>
<!-- 10st line -->
<TR>
    <! right border line>
    <TD bgcolor='#E6EFF8'>&nbsp;</TD>
</TR>
<!-- 11st line -->
<TR>
    <! bottom border line>
    <TD bgcolor='#4D7AB1' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#83A4CD' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#B6C7E1' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#E6EFF8' WIDTH=25%>&nbsp;</TD>
    <TD bgcolor='#E6EFF8'>&nbsp;</TD>
    <TD bgcolor='#E6EFF8'>&nbsp;</TD>
</TR>
</TABLE>

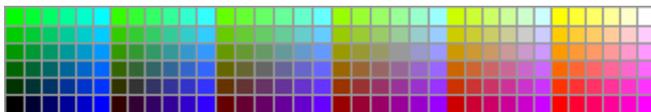
```

Restrictions

For a real web application, nobody would choose the layout above. There are some restrictions that should be considered if not only an Intranet with fixed defined HTTP browser and settings are used. The real world looks different.

Colors

Colors should be used from the so called "Netscape Color Cube" (see the figure below).



These are the 216 colors that the Netscape browser uses in its palette. By using these colors on your web pages, you can be assured that the viewer will see your images exactly as you intended. If you do not use the color cube, Netscape will use it for you.

Even if Internet Explorer with 256 colors is used, these colors work well. Other colors will lead to the same effects.

This happens not only to fix coded color setting inside your HTML, the same effect can be seen with pictures.

HTML 4.0 and Cascading Style Sheets

Cascading Style Sheets (CSS) and HTML 4.0 are the current standard of HTML. But only HTML 3.2 is really browser independent. With Internet Explorer 4 and Netscape 4, CSS is implemented, but there are differences.

Fixed Table Size

Our page uses fixed table cell sizes to display the background pictures in the right way. However, every browser allows you to set up the size of the font used.

It is very common that pages only look good if a specific font size is used. Other fonts lead to strange effects beginning with text that is displayed at the wrong column and leads to strange looking background pictures.

A table will always resize to display the text inside, regardless of whether a size is given. If more space is needed, the space is taken.

Therefore it is better to set up table cell only with percent settings.

Chosen Layout

The new layout, realized with a table, looks like the figure below.



Input Layout

It is necessary to return input data from the HTML page. Forms enable you to send data back to the HTTP server. Input elements are INPUT, TEXTAREA and SELECT (HTML 3.2). A form should have at least one submit button.

The main input screen to select a range of employees may look like this:

The HTML looks like this:

```
<FORM ACTION="/cgi-bin/nwwcgi.exe/sysweb/nat-env">
from:
<INPUT TYPE="text" NAME="FROM" VALUE="A" SIZE="5">
to:
<INPUT TYPE="text" NAME="TO" VALUE="Az" SIZE="5">
<INPUT TYPE=submit VALUE="Browse Employee">
</FORM>
```

The code shows that it is only possible to specify one URL to be called. However, the specification is supposed to call different selections. One way to do this is to add more than one form:

Another possibility is to call only one program, some kind of dispatcher. This program calls the subprogram needed. The dispatching can be done by the value of the submit button. Therefore, the submit button must have a name, e.g. "TODO". The form then looks like the figure below:

A form with a light green background. It contains two text input fields: 'from: A' and 'to: Az'. To the right of these fields are three buttons: 'Select', 'Browse', and 'Listing'.

If we now combine the given layout with our form, our main page will look like the figure below:

The screenshot shows a web page titled 'Maintenance of File' with a blue header. Below the header is a 'MENU' section. On the left, there is a vertical navigation bar with buttons for 'Home', 'Select', 'Browse', 'List', and 'New'. In the center, there is an illustration of a group of business professionals. To the right of the illustration, there is a form with a 'File:' dropdown menu set to 'EMPLOYEES', a 'Selection by Name' label, and two text input fields for 'From: A' and 'To: D'. At the bottom right, there is a 'Natural' logo.

Using the submit buttons does not lead to a harmonious display, because the appearance of the submit buttons cannot be changed.

However, submit buttons can be replaced by pictures, as in the figure below:

A form with a light green background. It features three text input fields on the left: 'Select...', 'Browse...', and 'Listing...'. To the right of these fields are two text input fields: 'from: A' and 'to: Az'.

However, the pictures do not look like submit buttons, or links to another object. Will they do something?

It would be better if a menu could be created with changing images. This cannot be done by using only HTML. If JavaScript can be used, it is possible to change the appearance of a picture if the mouse is over it:

```
script language="JavaScript" type="text/javascript" !-- function imgAct(imgName) { document[imgName].src =
"/pictures/nww_" + imgName + ".on.gif"; } function imgInact(imgName) { document[imgName].src =
"/pictures/nww_" + imgName + ".gif"; } !-- [FrontPage HTML Markup Component][FrontPage
Component]
```

See the code of the JavaScript:

```
<script language='JavaScript'><!--  
function imgAct(imgName)  
{  
  document[imgName].src = "/pictures/nww_" + imgName + "on.gif";  
}  
function imgInact(imgName)  
{  
  document[imgName].src = "/pictures/nww_" + imgName + ".gif";  
}  
!--></script>  
<a href='#name' onMouseOver='imgAct("brow")' onMouseOut='imgInact("brow")'><img alt='Browse'  
name='brow' src='/pictures/nww_brow.gif' border='0' width='129' height='21'></A>
```

There are two parts:

- the **functions** changing the source of an image and
- the **call** of the different functions.

If scripting is not allowed, only the up picture can be seen. This is because every browser ignores unknown tags. The coding cannot be seen, because it is hidden inside a comment. The closing HTML comment has to start with an JavaScript comment!

If scripting is allowed, the script is loaded. It is necessary to name the img tag to access this image later on. If now the mouse is dragged over the image, the onMouseOver event occurs and the function imgAct("brow") is executed. This function now replaces the source of the image "brow" with the image "/pictures/nww_" + "brow" + "on.gif". If the mouse leaves the image, an onMouseOut event is sent and imgInact("brow") is executed. Now the picture "/pictures/nww_" + "brow" + "on.gif" will be loaded.

As the example shows, two pictures for the different state of the button are needed. The name of the image is generated from the img tag name and an individual extension.

One question remains: How does this work together with a form?

JavaScript offers the possibility of calling a method that behaves like a pressed submit button. An anchor can call a function instead of linking to to another URL. Using this leads to the following script:

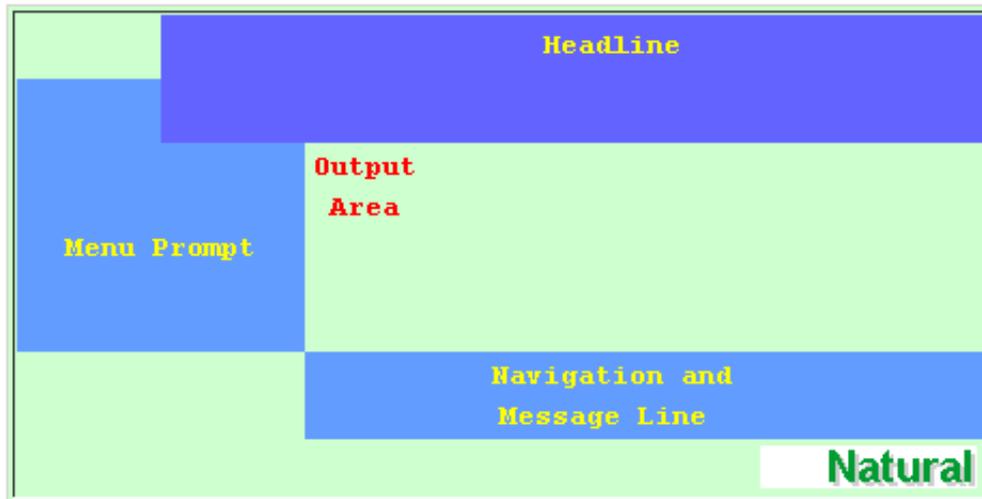
```
<script language='JavaScript'><!--
function goon()
{
  document.FMENU.submit();
}
function imgAct(imgName)
{
  document[imgName].src = "/pictures/nww_" + imgName + "on.gif";
}
function imgInact(imgName)
{
  document[imgName].src = "/pictures/nww_" + imgName + ".gif";
}
//--></SCRIPT>
<form method='get' action='/cgi-bin/nwwcgi.exe/sysweb/nat-env ' name='FMENU'>
<a href='javascript:goon()' onMouseOver='imgAct("brow")' onMouseOut='imgInact("brow")'><img
alt='Browse' name='brow' src='/pictures/nww_brow.gif' border='0' width='129' height='21'></a>
</form>
```

Again the form tag has to be **named** to access later. The **href** attribute of the anchor now calls the function **goon()** which calls the **submit()** method of the form tag.

Now putting everything together, our layout for the main page looks like this:

<h1>Maintenance of File</h1>	
<p>MENU</p>	
<p>Back ← Home... Select... Browse... Listing... New...</p>	<p>Data Source</p> <p>File: <input type="text" value="EMPLOYEES"/></p>
	<p>Selection by Name</p> <p>From: <input type="text" value="A"/></p> <p>To: <input type="text" value="D"/></p>
	
<p style="text-align: right;">Natural</p>	

Functional Parts



How to Use Frames

It is easy to set up frames, but frames cannot be bookmarked. It is possible to bookmark subpages of the frame, but you cannot start up the whole frame again.

Instead of frames, tables can be used for page formatting. The different parts of a page can be generated by separate subprograms. The resulting page is usually compatible with the frame-based page.

How to Use JavaScript

Using JavaScript on your page can be very useful. Use special script files and link these files to your application with the "SRC" attribute of the script tag.

Scripting on mainframes can cause specific problems. The character [] { } may not be defined in the EBCDIC character set used or cannot be typed on the terminal used.

If you use the Natural Web Server Extensions, a special feature can translate the &#.; equivalent back to the characters before sending to the HTTP server.

Do not forget to specify a <NOSCRIPT> section for older browsers.

If scripting is not allowed, you may not allow the user to go on or you may specify alternative pages.

Web Interface Program Design

The design of the web application is restricted by the web technologies used. It is possible to find this or that work around but this is never a general solution. Using another browser or different versions of the same browser can have significant effects. Setting up browser security to a different level can change the application's environment.

This section covers the following topics:

- Data Input
 - Global Data
 - Dispatch
 - External Data
 - Modular Pages
 - Chosen Program Design
-

Data Input

All data given by the HTTP server and web browser must be treated as unchecked data! Nobody can force a user to use scripting, given colors or other settings.

The application has to check all input data again for consistency, even if the calling page contains scripts for validation. Not only the value of the given data is unchecked, even the data format and length is insecure. An HTML page always can be copied with "Save as" and then modified for your own needs.

Use the Natural IS Option for checking format and length of values. The example below shows reading of the value START followed by checking whether the value is an N5 value.

```
PERFORM W3READ-ENVIRONMENT "START" "P" W3VALUE W3MAX
IF W3MAX GT 0 THEN
  IF W3VALUE-A5 IS (N5) THEN
    #DEMO-PARM.#START:= VAL(W3VALUE-A5)
  ELSE
    #DEMO-PARM.#START:= 1
  END-IF
ELSE
  #DEMO-PARM.#START:= 1
END-IF
```

Global Data

With your web application, you have different ways of saving global data.

1. Cookies

Cookies are data saved with your local browser. The number of cookies is limited. Use only one cookie for your whole application, to save different data needed for your application.

Cookies are delivered with the variable HTTP_COOKIE. The Natural Web Server Extension used has to be advised to pass cookies to your application. Your initialization file must contain the following line:

```
ENV=HTTP_COOKIE
```

Inside your program, cookies can be read with the normal functionality:

```
PERFORM W3READ-ENVIRONMENT "HTTP_COOKIE" 'S' W3VALUE W3MAX
IF W3MAX > 0 THEN
  #MY-COOKIE := W3VALUE
ELSE
  RESET #MY-COOKIE
END-IF
```

Add your cookie, which has to be set with an expire time, at the return page:

```
ADD 1 TO #NUMBER
COMPRESS "COUNTER=" #NUMBER ";" INTO COOKIE-NAME LEAVING NO
COMPRESS 'Set-Cookie:' COOKIE-NAME
  'expires=Wednesday, 09-Nov-99 23:12:40 GMT' ##HTTP_NEWLINE_END
  INTO W3VALUE
PERFORM W3HTTP W3VALUE
```

2. Hidden Input Fields and Additional URL Parameters

If your page contains a form, additional data can be saved as a hidden input field. This data will be sent if the form is submitted, but cannot be seen when the HTML page is displayed. If not form tag is used, it is possible to add this parameter direct to the URL of the next pages called.

Setting of "global data" using hidden input fields inside a form:

```
PERFORM H3-INPUT "H" "START" #START 0 0
PERFORM H3-INPUT "H" "FROM" #FROM 0 0
PERFORM H3-INPUT "H" "TO" #TO 0 0
```

3. Data Saved on the Server Side (in a Database)

It is very common to save data for a given user at the application database. This is only useful if password saved applications are used. The username then can be used to load specific data out of the database.

Dispatch

For many applications, it is common that for a given input, different actions can be called. However, submitting a form tag can only call one program.

One way of solving the problem is using scripting and changing the URL of the called program depending on the submit button pressed.

A second way is to create one dispatcher program that evaluates which program should be called, depending on the pressed submit button.

A combination of both can also be done. Then the submit can be realized by a normal link that starts the submit. The different ways can be selected by changing the value of a hidden input field.

External Data

If pictures or other external static data from the server is needed, use a dynamically created URL instead of a static one. Use an environment variable, set by the Natural Web Server Extension to specify a non-standard path.

Setup your own variable for pictures in the Natural Web Server Extension initialization file:

```
SETENV=pictures:=/gif
```

Modular Pages

For a complex application, it is useful to define subroutines for special parts of the generated page.

- Page design split input: head, body, menu ...
- Use external subroutines.

Example of a modular generation page:

```

* Check given parameter of mandatory value is set.
IF #DEMO-PARM.#PERSON EQ " " THEN
  #DEMO-PARM.#MSG := 'Please Select a person!'
  INCLUDE D4-BACK
END-IF
*
* Specify the name of this page
#DEMO-PARM.#TODO := "SHOW"
#DEMO-PARM.#CAPTION := 'Show'
*
* Define the Navigationbar items to be shown
#DEMO-MENU.#ME := "T" /* Menu
#DEMO-MENU.#CH := "T" /* Change
#DEMO-MENU.#DE := "T" /* Delete
*
* Define the Navigationbar back to Select/Browse
PERFORM D4-BACKB #DEMO-MENU #DEMO-PARM
*
* Generate Start of HTML page and Navigationbar
PERFORM D4-START #DEMO-MENU #DEMO-PARM
*
* add global ID as hidden input field
PERFORM D4-HPERS #DEMO-MENU #DEMO-PARM
*
* Read Employee data from the Database
CALLNAT "D4EMSHOW" #DEMO-PARM.#PERSON #OUT-P-NAME
          #OUT-P-FIRST-NAME #OUT-P-BIRTH #OUT-P-SEX
          #OUT-P-DEPT #OUT-P-CITY #OUT-P-AREA-CODE
          #OUT-P-ZIP #OUT-P-COUNTRY #OUT-P-PHONE
          #OUT-P-PICTURE #DEMO-PARM.#MSG
*
* Generate HTML output for this page
INCLUDE D4-DISPL
*
* Generate end of HTML page
PERFORM D4-END #DEMO-MENU #DEMO-PARM
END-SUBROUTINE

```

Chosen Program Design

The application uses one startup page and one dispatch page. These pages contain a form which contains all necessary input fields. Data will be passed as hidden fields from one page to another. The different functions of the page will be selected by the value of a specific field. The value will be set via a JavaScript program used on this page. This makes it possible to create a cursor-sensitive selection menu.

The main programs called for every page are the generation of the page head, the menu and the page body. The page head and body contain the layout information for the table structure used on the page. Data given from the previous page will be read to a parameter data area that will be passed to all subroutines used. This parameter data area contains an array to set up the relevant parts of the menu.

The layout of the pages is encapsulated with copycode if the layout is used more than once. Access to the database is separated to separate subprograms.

External data as gifs and the JavaScript source can be reallocated using variables defined in the Natural Web Server Extension.

Functional Parts

Main Programs

D4ENTER, Subprogram
entrance tunnel
D4MENU, Subprogram
main menu
D4CHOOSE, Subprogram
dispatcher program

Database Access

D4EMBROW Subroutine
browse
D4EMCHAN Subprogram
change
D4EMDEL, Subprogram
delete
D4EMLIST, Subroutine
list
D4EMNEW, Subprogram
add new
D4EMSEL, Subroutine
select
D4EMSHOW, Subprogram
show

General HTML Layout

- D4-BACK, Copycode
 - relocate if nothing is selected
- D4-BACKB, Subroutine
 - back button to return correct to list/browse
- D4-DISPL, Copycode
 - Display Screen for Show/Delete
- D4-END, Subroutine
 - generate the "end" of a demo HTML page
- D4-HGLOB, Subroutine
 - write "global" variables as `START`, `FROM`, `TO`, `OT`, `STATUS` to the HTML page
- D4-HPERS, Subroutine
 - write the "global" variable `Person "ID"` to the HTML page
- D4-IPAGE, Copycode
 - add a person
- D4-MENUB, Subroutine
 - generate the menu bar for the demo HTML page
- D4-MODIF, Copycode
 - input screen for change/new
- D4-MSG, Subroutine
 - generate the message line
- D4-PARM, Parameter Data Area
 - cross application data
- D4-RGLOB, Subroutine
 - read variables send by the previous page
- D4-START, Subroutine
 - generate the "start" of a demo HTML page

Sub HTML-Pages

- D4BROWSE, Subroutine
 - browse persons
- D4CHANGE, Subroutine
 - change a person
- D4DELETE, Subroutine
 - delete a person
- D4LIST, Subroutine
 - list persons
- D4NEW, Subroutine
 - add a person
- D4SELECT, Subroutine
 - select a person
- D4SHOW, Subroutine
 - show a person

Special Purpose

D4NOTIMP, Text

static page for not implemented

D4TEMPEL, Subprogram

display static pages saved as Natural text members

Web Interface Implementation

This section covers the following topics:

- HTML to Natural
 - Reuse of Global Parts
 - Use of w3text versus w3html
 - Increased Performance using w3text / w3html
-

HTML to Natural

With common HTML editors, it is very easy to specify an HTML page layout. Now, using Natural Web Interface, this layout must be transferred to Natural.

This can be done with a utility called HTML to Natural that is delivered with the Windows NT platform. This utility allows you to generate a Natural subprogram, containing all necessary interfaces, that will generate the specified page.

Reuse of Global Parts

Some parts of the application need to be replicated on every page. Use Natural copycode or external subroutines for partial generation.

Use of w3text versus w3html

If your text does not contain special characters or the string contains HTML tags, use W3TEXT. Otherwise, you can use W3HTML to translate unsaved characters, such as < > & ? into an HTML-conform equivalent.

The same happens to Natural characters, such as the German ü, ä and ö or special Spanish and French characters, such as é, è and ê.

The translation of these characters decreases the speed of the application.

Increased Performance using w3text/w3html

As Natural only works with fixed length strings, every string is filled with spaces. For HTML, spaces are not relevant, because if more than one white space occurs, it will be compressed to only one. Returning pages with a lot of spaces does not alter the result of the rendered HTML page. But the numbers of bytes to be transferred increase.

The output routines of the Web Interface will strip all trailing blanks before sending the string back. Therefore the string has to be scanned beginning from the end of the string and then the last space has to be found.

To increase the speed of your application, use long strings and/or terminate the string with the string defined at the variables ##HTTP_END and ##HTTP_NEWLINE_END from the parameter data area W3CONST.

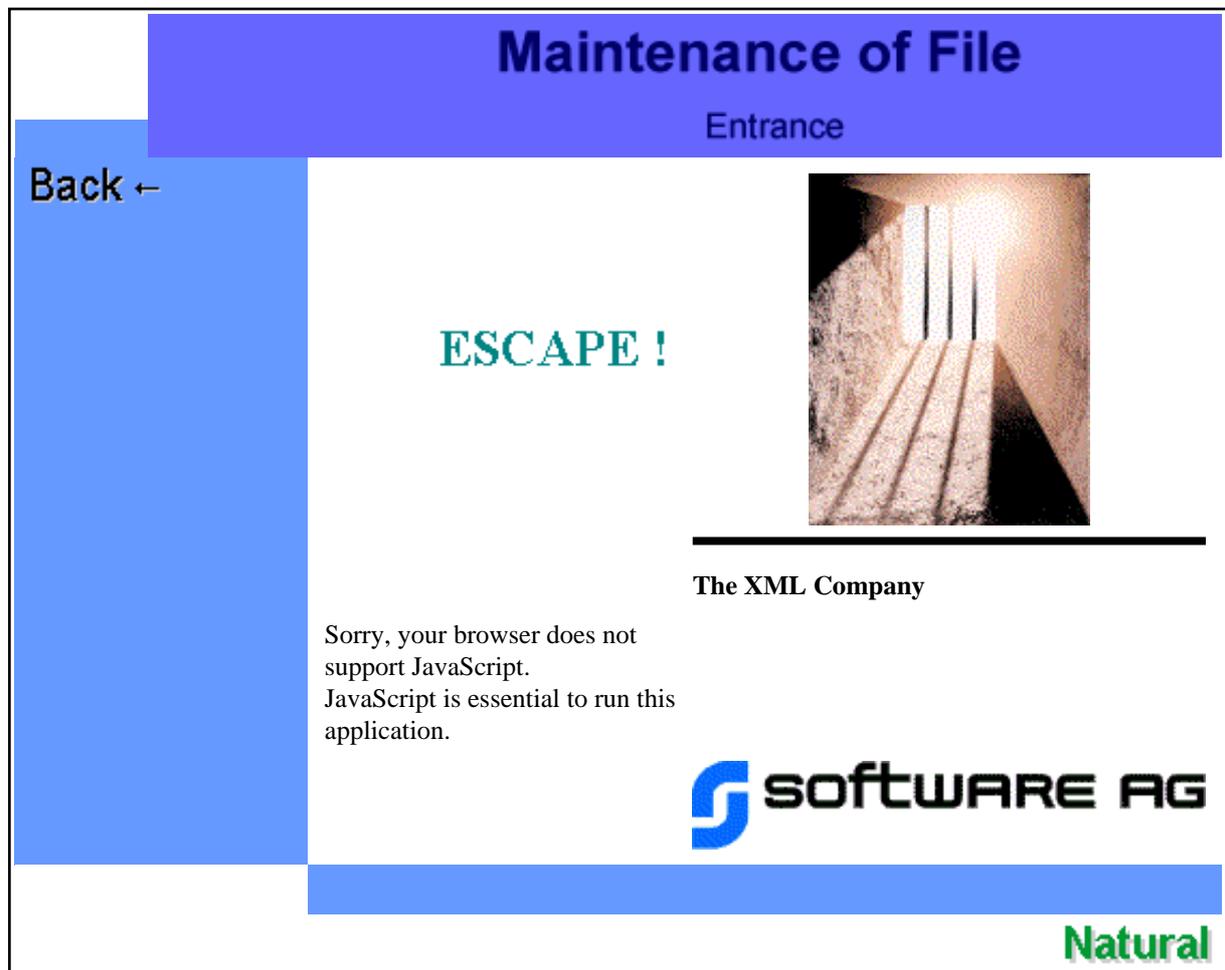
Web Interface Fine Tuning

This section covers the Entrance Tunnel of the Web Interface.

Entrance Tunnel

If your program needs specific versions of HTML browser, or JavaScript as prerequisite, do not let the user run on an incomplete page or get error messages.

A common technique for browser-specific checks is a so-called entrance tunnel. This is the first page of your application. The page can show the logo of your application or just a link to your application. Now the next pages can be browser-dependent, or you may not allow the user to enter:



There are two places to check such specifics:

- Server-side checking at program generation of your dynamic page.
- Browser-side scripting.

The version of an HTML browser used can be checked if the variable `HTTP_USER_AGENT` is read.

With the Natural Web Interface, use the following lines of code:

```

PERFORM W3READ-ENVIRONMENT H3NAME H3SERVER H3VALUE H3VALUE-MAX
*
IS-IE4 := FALSE
IS-NAV4 := FALSE
*
IF H3VALUE-MAX GT 0 THEN
  EXAMINE H3VALUE FOR "Mozilla" GIVING INDEX II
  IF II EQ 1 THEN
    EXAMINE H3VALUE FOR "MSIE" GIVING INDEX IJ
    IF IJ GT 0 THEN
      IS-IE := TRUE
    ELSE
      IS-NAV := TRUE
    END-IF
  END-IF
END-IF
*
* Browser Dependent Coding
* only if Navigator is used.
IF IS-NAV EQ TRUE THEN
...

```

The following value identifies a Microsoft Internet Explorer 4.01:
Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)

With the following value a Netscape Communicator 4.5 will respond:
Mozilla/4.5 [en] (WinNT; I) .

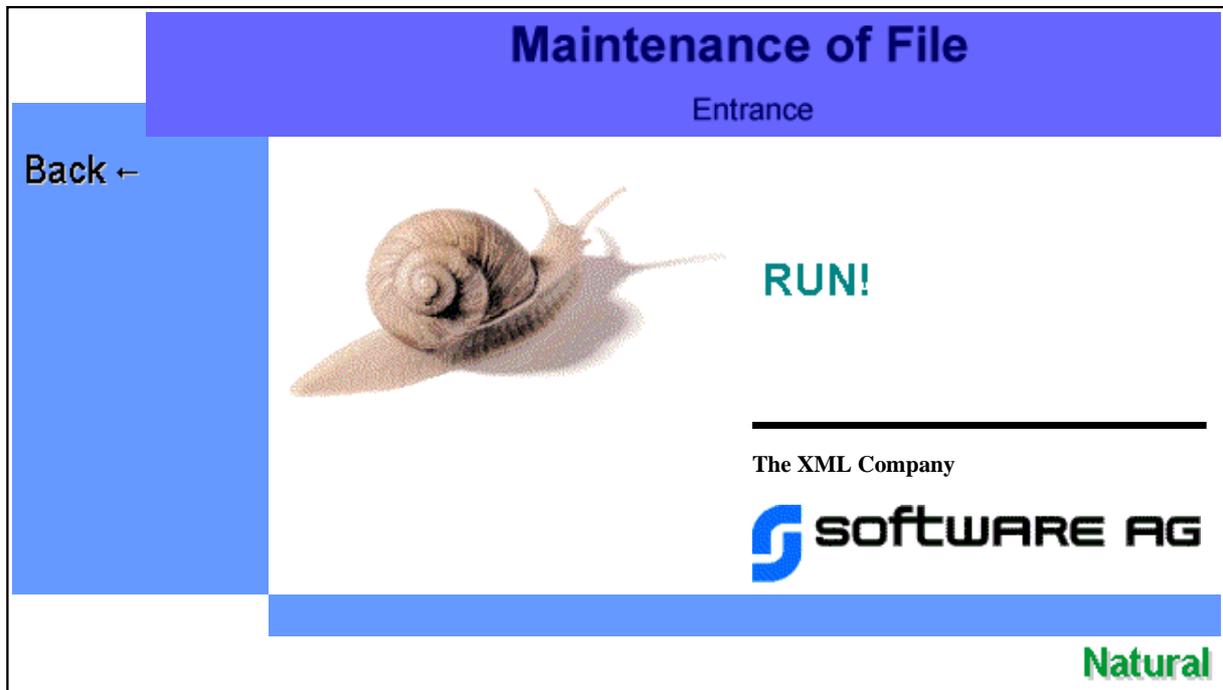
The same check can be done with an Java program on the client side.

```

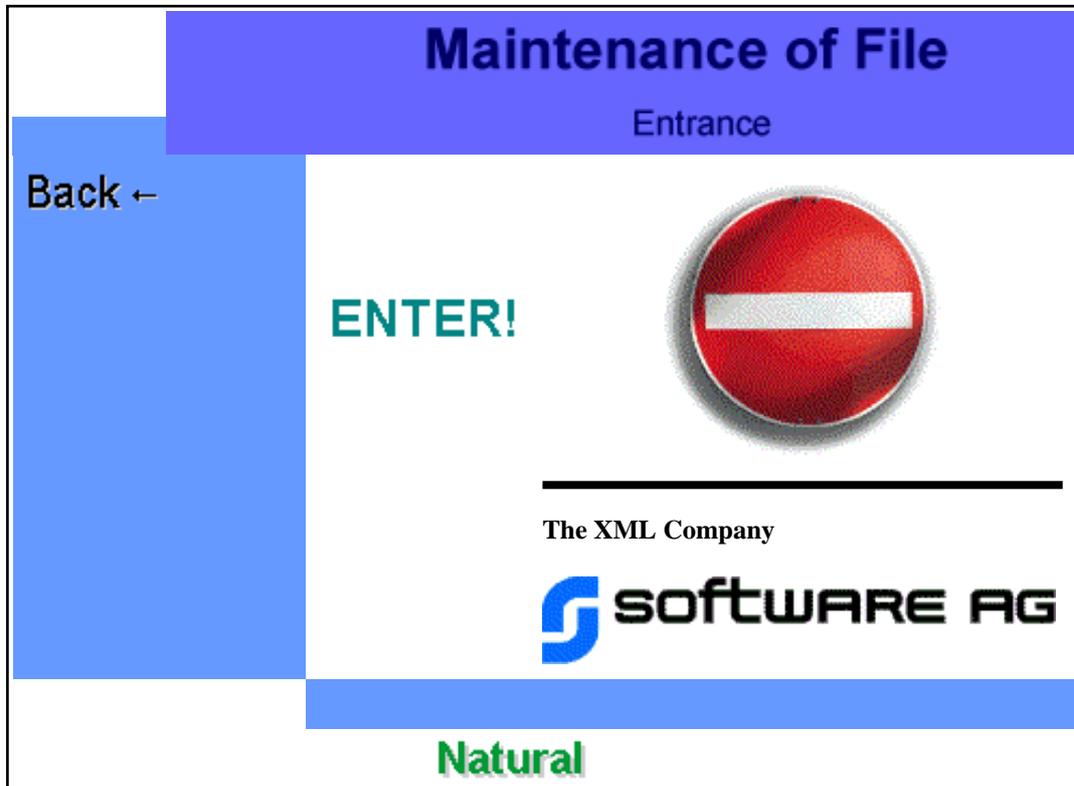
// Check for used Browser
function Is ()
{ // convert all characters to lowercase to simplify testing
var agt=navigator.userAgent.toLowerCase()
// *** BROWSER VERSION ***
  this.major = parseInt(navigator.appVersion);
  this.nav = ((agt.indexOf('mozilla')!=-1) && ((agt.indexOf('spoofer')== -1)
&& (agt.indexOf('compatible') == -1)))
  this.ie = (agt.indexOf("msie") != -1)
}
// Browser Dependent Coding
// only if Navigator is used.
IF is.nav { ...

```

If your application should not look the same all the time, you can alter your entrance tunnel:



or:



Natural Web Server Extensions - Overview

This document comprises the following sections:

- Introduction
First you can get an insight into the working and installation procedures of the Natural Web Server Extensions.
- Initialization File
Here we describe parameters and variables of the initialization file.
- Configuring the Natural Web Interface
Here you can find information needed to configure the Natural Web Interface.
- Error Messages
Here we list likely errors.

Natural Web Server Extensions - Introduction

This section covers the following topics:

- General Information
 - Installation - RPC / DCOM
 - Transformations
 - Variables
 - Error Logging and Messages
 - Calling Programs
-

General Information

The Natural Web Server Extensions part is basically a program called from an HTTP server. The Natural Web Server Extensions takes parameters, given by the HTTP server, repackages them and performs a broker RPC or a DCOM call to the requested Natural program using a standard parameter data area. Calls are transmitted by the EntireX Broker or DCOM.

As of Version 4.1, three HTTP Server interfaces will be supported:

- Common Gateway Interface (CGI), for supported server and platforms,
- Internet Server Application Programming Interface (ISAPI) only for Microsoft Internet Information Server on Windows NT.
- Netscape Server Application Programming Interface (NSAPI) only for Netscape FastTrack Server.

Installation - RPC / DCOM

Each Natural Web Server Extension consists of two files:

- an executable and
- an initialization file.

These files can be renamed. The initialization file has the same name as the executable file, but with the extension .ini. The two files must be in the same directory.

Copy the files to appropriate locations of the web server, or parameterize the web server so that it accesses the files direct.

	RPC	DCOM
CGI	nwwcgi.exe nwwcgi.ini	nwwdcgi.exe nwwdcgi.ini
ISAPI	nwwisapi.dll nwwisapi.ini	nwwdisapi.dll nwwdisapi.ini
NSAPI	nwwnsapi.dll nww/nsapi	nwwdnsapi.dll nwwd/nsapi
Parameters	RPC_ETB_ID_NAME = broker name RPC_SERVER_NAME = service name NWW_INOUT_LENGTH = amount of transferred data	NWW_INOUT_LENGTH = amount of transferred data

Note:

Some HTTP servers allow executable files without the extension .exe. This means that executables with and without the .exe extension are possible.

Transformations

Parameters sent by the HTTP server via the interface are given by means of specific variables or a transfer area. User data contained in a transfer area or the variable QUERY_STRING will be recognized and preprocessed. In particular, the encoding of the URL will be undone.

The design of the Natural Web Server Extensions allows only the transmission of non-binary data, because the data is converted from ASCII to EBCDIC and vice-versa if needed.

Variables

Only variables specified on your HTML page will be automatically transferred to your called program. Other variables available from the HTTP server must be specified.

Each variable to be transferred needs an entry in the initialization file.

It is also possible to add variables that will be treated as system environment variables.

Error Logging and Messages

You can set up your own error screen with a specific HTML page. Variables of the environment can be specified in this error page.

The page last transferred can be copied to a file and errors can be written to an error log file.

Calling Programs

To call a program from your browser, you have to specify a uniform resource locator (URL) which contains the name of you HTTP server and the name of a cgi-enabled directory, where you have copied the files of the Natural Web Server Extension. Then you have to specify the Natural Web Server Extension program name followed by a Natural library and a subprogram name.

	URL for RPC	URL for DCOM
CGI	<i>http://server-name/cgi-library/nwvcgi.exe/your-library/your-program</i>	<i>http://server-name/cgi-library/nwvdcgi.exe/your-library/your-program</i>
ISAPI	<i>http://server-name/cgi-library/nwwisapi.dll/your-library/your-program</i>	<i>http://server-name/cgi-library/nwvdisapi.dll/your-library/your-program</i>
NSAPI	<i>http://server-name/nww/nsapi/your-library/your-program</i>	<i>http://server-name/nwv/nsapi/your-library/your-program</i>

Natural Web Server Extensions - Initialization File

This section covers the following topics:

- General Information
 - RPC Parameters
 - DCOM Parameters
 - Natural Web Server Extension Settings
 - HTTP Server Variables
 - Additional Variables
 - Error Templates
-

General Information

The Natural Web Server Extension takes runtime parameters from an initialization file. The executable file looks for an initialization file with the same name and extension .ini in the current working directory.

The names of the variables are not case sensitive, as all variables used on the WWW. Variables are limited to 72 characters; blanks are recognized as normal characters, so parameters can be specified multiple times.

RPC Parameters

These parameters are needed for communication with EntireX RPC.

Parameter	Description
RPC_CLASS_NAME	Defines the class of the service used. Always use RPC.
RPC_ETB_ID_NAME	Name of the EntireX Broker to be called.
RPC_NO_LOGON	Logon to the library specified at the URL. Default is 0.
RPC_SERVER_NAME	Name of the called Broker Service.
RPC_SERVICE_NAME	Defines the called service. Always use CALLNAT.
RPC_TIME_OUT	Defines the timeout for the call. Default is 7000.

DCOM Parameters

These parameters are needed for the communication with DCOM.

Parameter	Description
DCOM_SERVER_NAME	Name of the called DCOM Server. (Specify only if the Natural Server is not running on the same computer.)

Natural Web Server Extension Settings

This group of parameters define the settings of the Natural Web Server Extension.

Parameter	Description
ECHO_ENVIRONMENT	This parameter is only useful if the default error page is used. If this parameter is specified and set to 1, equal to the \$NWW_ENVIRONMENT of the user-defined error page, all environment variables will be written as comment lines to the error page.
ERROR_LOG_FILE	Defines a file for error logging. If this parameter is not specified, the log is disabled. Each log entry has the same layout and can easily be located in the error-log file by searching for the string the cgi is named. Sample Log Entry: [Thu Jun 28 10:51:19 2001] nwwcgi.exe 04.02.00 Win32: processing of /cgi-bin/nwwcgi.exe failed for Lib:{library} Sub:{subprogram} Path:{path_info}, for natweb.software-ag.de reason NWW0001 No subprogram and library specified.
ERROR_STDERR	If this parameter is set to 1, all errors are logged via stderr. The actual location of the log file depends on the HTTP server used and the way it has been parameterized. See also ERROR_LOG_FILE. Some HTTP servers do not support the use of stderr.
ERROR_TEMPLATE	Defines an error template file. If this parameter is not specified, a default error page will be generated. See Error Templates below.
NWW_INOUT_LENGTH	Defines the amount of the transferred data. This parameter defines the dimension of the parameter Out_Page of the IDL file. Used IDL File: <pre>DEFINE DATA PARAMETER 1 Version-Nr (A15) In 1 Log-Time (A30) In 1 Out_Page (A RPC_INOUT_FORMAT 1:RPC_INOUT_LENGTH) In Out 1 Out_Page_Count(I04) In Out 1 Result (I04) Out END-DEFINE</pre>
NWW_PASSWORD	Defines the password for the user ID.

<p>NWW_PATH_INFO</p>	<p>To test the Natural Web Server Extension in stand-alone mode (test environment), set this parameter to specify the library and program name. If you use the Natural Web Server Extension in the regular mode (with HTTP-Server) you must disable this parameter.</p> <p>Example: <code>NWW_PATH_INFO=/syshtml/nat-env</code></p>																		
<p>NWW_PATHINFO_PREFIX</p>	<p>This parameter can only be used in conjunction with the ISAPI interface. If the interface is defined as application mapping (e.g. for directory nww and the extension .nww), the PATH_INFO variable delivers a prefixed URL with directory and the file name (e.g./nww/my.nww/sysweb/nat-env). This prefix (/nww/my.nww) has to be removed. Use this parameter to the specified used prefix.</p> <p>Example: <code>NWW_PATHINFO_PREFIX=/nww/my.nww</code></p>																		
<p>NWW_OUT_CSS</p>	<p>Replaces the strings by the specific characters:</p> <table border="1" data-bbox="584 862 772 1207"> <thead> <tr> <th>String</th> <th>Character</th> </tr> </thead> <tbody> <tr> <td>&#09;</td> <td>--> (Tab)</td> </tr> <tr> <td>&#64;</td> <td>@</td> </tr> <tr> <td>&#91;</td> <td>[</td> </tr> <tr> <td>&#92;</td> <td>\</td> </tr> <tr> <td>&#93;</td> <td>]</td> </tr> <tr> <td>&#123;</td> <td>{</td> </tr> <tr> <td>&#124;</td> <td>/</td> </tr> <tr> <td>&#125;</td> <td>}</td> </tr> </tbody> </table> <p>This setting can be useful if cascading style sheets are used and the RPC server is placed on a computer which uses the EBCDIC code. Default is 0. Use 1 to activate.</p>	String	Character			--> (Tab)	@	@	[[\	\]]	{	{	|	/	}	}
String	Character																		
		--> (Tab)																		
@	@																		
[[
\	\																		
]]																		
{	{																		
|	/																		
}	}																		
<p>NWW_OUT_CSS_TRANSLATE</p>	<p>Replaces the characters by the specific hex values : (Default value for ASCII)</p> <table border="1" data-bbox="584 1444 807 1792"> <thead> <tr> <th>Character</th> <th>Hex value</th> </tr> </thead> <tbody> <tr> <td>--> (Tab)</td> <td>09</td> </tr> <tr> <td>@</td> <td>40</td> </tr> <tr> <td>[</td> <td>5B</td> </tr> <tr> <td>\</td> <td>5C</td> </tr> <tr> <td>]</td> <td>5D</td> </tr> <tr> <td>{</td> <td>7B</td> </tr> <tr> <td> </td> <td>7C</td> </tr> <tr> <td>}</td> <td>7D</td> </tr> </tbody> </table> <p>Example for english EBCDIC (codepage 37): # (tab), @, [, \,], {, , } <code>NWW_OUT_CSS_TRANSLATE=05,7C,AD,61,BD,C0,4F, D0</code></p>	Character	Hex value	--> (Tab)	09	@	40	[5B	\	5C]	5D	{	7B		7C	}	7D
Character	Hex value																		
--> (Tab)	09																		
@	40																		
[5B																		
\	5C																		
]	5D																		
{	7B																		
	7C																		
}	7D																		
<p>NWW_USER_ID</p>	<p>User ID used for the RPC.</p>																		
<p>NWW_RETRY</p>	<p>If a 3009 error (NAT3009 Transaction aborted) occurs, this parameter defines how often the program will be called again. Default is 3.</p>																		

INI_RELOAD	Load initialization file only once during the first call. Not for CGI interface. Default is 1.
REMOVE_USER_DOMAIN	IIS server on NT deliver as REMOTE_USER the username prefixed with the name of the domain the user belongs to. Natural can only handle user names with a maximum length of 8 characters. If USE_REMOTE_USER is set to 1 and REMOVE_USER_DOMAIN is set also to 1 the used domain name from the given REMOTE_USER name is removed. This means the information after the last "/" is delivered to Natural as the user name .
TRACE_FILE	If a file name is specified, the last pages returned to the HTTP server will be saved to this file. If this parameter is specified, no output is written.
USE_REMOTE_USER	Replace the RPC_USER_ID with the given REMOTE_USER. Set to 1 to activate it.

HTTP Server Variables

All HTTP server variables to be transferred to the called program must be specified. To do this, specify the variable ENV with the name of the variable to be transferred. The ENV variable can be specified multiple times.

Some useful variables:

```
ENV=REMOTE_HOST
ENV=REMOTE_ADDR
ENV=SCRIPT_NAME
ENV=HTTP_REFERER
ENV=HTTP_HOST
ENV=HTTP_COOKIE
```

For more information on variables, see <http://hoohoo.ncsa.uiuc.edu/cgi/env.html>.

Additional Variables

With the Natural Web Server Extension, it is additionally possible to transfer variables to the called program. To do this, specify the variable SETENV with the name of the variable followed by := and the value to be transferred. The SETENV variable can be specified multiple times.

Example:

```
SETENV=PICTURES:=/pictures
```

Error Templates

Default Error Report

If parameter ERROR TEMPLATE is not specified, a default is used.

This is an example of a default error report:

nwwcgi.exe Error Report

Natural Web Interface NWW5100c Win32

The following error has been logged in the error log file:

/cgi-bin/nwwcgi.exe: processing of subprogram/method **NAT-INFO**
at library/class **SYSWEB** failed.

reason: **NWW0011** ERX error 80010014 occurred.

Severity = Error
Facility = 65536
Returncode = 20
Subfacility = 3
Location = 0

Message:
ERX_E_SERVICE_NOT_AVAILABLE - ETB error code 02150148

for: pcnatweb.software-ag.de:80

path: /sysweb/nat-info

NWW Error - Fri Mar 15 10:20:28 2002[Natural](#)

Specifying your own Error Template

You can also specify your own error template. The error template is basically a normal return page. As for all return pages, the content type must be set. The only addition is the replacement of variables. To do this, specify the environment variable beginning with a \$ sign. See Example of an Error Template below.

The following "environment variables" are additionally available for error templates:

Environment Variable	Description
NWW_LOGTIME	Time and date the error will be logged if an ERROR_LOG_FILE is specified.
NWW_VERSION	Version number of the Natural Web Server Extension.
NWW_RUN	Name of the program that was called.
NWW_ERROR	Number of the error that has occurred.
NWW_LIBRARY NWW_CLASS	Name of the library/class that was called.
NWW_SUBPROGRAM NWW_METHOD	Name of the subprogram/method that was called.
NWW_ENVIRONMENT	All environment variables will be written as comment lines to the error page.

Example of an Error Template

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <TITLE>$NWW_RUN Error Report - $NWW_LOGTIME</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF" text="#000000">
<TABLE border="0" width="100%" cellpadding="5">
  <TR bgcolor="#CCFFCC">
    <TD align="center">
      <H2 align="center">
        $NWW_RUN Error Report
      </H2>
      <P align="center">
        <I><SMALL>Natural Web Server Extension Interface: $NWW_VERSION</SMALL></I></TD>
    </TR>
    <TR>
      <TD align="center"><B>The following error has been logged in the error log file:</B></TD>
    </TR>
  </TABLE>
  <TABLE border="0" width="100%" cellpadding="15" cellspacing="0">
    <TR valign="top">
      <TD align="right"><B>$SCRIPT_NAME:</B></TD>
      <TD align="left"><TT>processing of subprogram/method <B>$RPC_SUBPROGRAM</B><BR>
        at library/class <B>$RPC_LIBRARY</B> failed.</TT></TD>
    </TR>
    <TR valign="top">
      <TD align="right"><B>reason:</B></TD>
      <TD align="left"><PRE>$RPC_ERROR
  </PRE>
    </TD>
  </TR>
    <TR valign="top">
      <TD align="right"><B>for:</B></TD>
      <TD align="left"><TT>$SERVER_NAME:$SERVER_PORT</TT></TD>
    </TR>
    <TR valign="top">
      <TD align="right"><B>path:</B></TD>
      <TD align="left"><TT>$PATH_INFO</TT></TD>
    </TR>
  </TABLE>
  <TABLE border="0" width="100%" cellpadding="5">
    <TR bgcolor="#CCFFCC">
      <TD align="center">NWW Error Template - $NWW_LOGTIME</TD>
      <TD align="right">Natural</TD>
    </TR>
  </TABLE>
  <P>
  $NWW_ENVIRONMENT
</BODY></HTML>

```

Natural Web Server Extensions - Error Messages

This section lists error messages you may receive when working with the Natural Web Server Extensions.

Error Number	Error Message	Description	User	Programmer	Administrator
NWW0001	No library and subprogram specified.	The specified URL is not correct. Names of library and subprogram are missing.	Correct the URL.	None.	None.
NWW0002	No library specified.	The specified URL is not correct.	Correct the URL.	None.	None.
NWW0003	File ... not found.	The initialization file for your adapter cannot be found.	None.	None.	Check your installation.
NWW0004	No subprogram specified.	The specified URL is not correct.	Correct the URL.	None.	None.
NWW0010	RPC call failed.	EntireX RPC cannot be initialized.	None.	None.	Check installation.
NWW0011	ERX error ... occurred ...	Internal ERX error. See EntireX documentation for further information. If the error contains the following part: <i>Message:</i> ... <i>Program = NATSRVD</i> ... <i>Error = 00082</i> ... The called program does not work.	Correct URL.	Check and stow your program.	Check installation.
NWW0012	ERX error register.	EntireX RPC Service cannot be initialized.	None.	None.	Check configuration.
NWW0013	erx.dll cannot be loaded. Subcode:	EntireX erx.dll not found.	None.	None.	Check installation.
NWW0014	ERX logon failed.	EntireX logon cannot be performed.	Check User-ID, Password	Check installation file for Check User-ID, Password.	Check installation.
NWW0015	ERX logoff failed.	Logoff form EntireX failed.	None.	None.	Contact Software AG.
NWW0033	File ... not found (Error: ...).	The initialization file for your adapter cannot be found.	None.	None.	Check your obj.conf setup for NSAPI.
NWW0034	NWW_USER_ID too long.	User ID only with a maximum of 8 characters allowed.	None.	None.	Specify only user IDs with 8 characters or fewer, even if other system will allow more.
NWW0035	NWW_PASSWORD too long.	Passwords only with a maximum of 8 characters allowed.	None.	None.	Specify only (user-) passwords with 8 characters or fewer, even if other system will allow more.
NWW0036	Natural Library Name too long.	Natural allows only library names up to 8 characters.	Check URL.	Check URL specification.	None.
NWW0037	Natural Subprogram Name too long.	Natural allows only Subprogram names up to 8 characters.	Check URL.	Check URL specification.	None.

NWW0099	CONTENT_TYPE: ... is not supported.	Only data with CONTENT_TYPE = application/x-www-form-urlencoded is supported.	None.	Do not use the attribute ENCTYPE at your FORM tag.	None.
NWW0100	RPC_INOUT_LENGTH is greater then 30000.	The output returned to the HTTP server is limited to restrictions of Natural RPC.	None.	None.	Change configuration
NWW0200	No Header specified.	Each page needs a header section at the return page.	None.	Each page should contain a CONTENT_TYPE. The header section has to be separated from the data by a blank line.	None.
NWW0201	Page contains no Data.	Every return page has to contain data.	None.	Correct the program.	None.
NWW1001	No class and method specified.	The specified URL is not correct. Names of class and method are missing.	Correct the URL.	None.	None.
NWW1002	No class specified.	The specified URL is not correct.	Correct the URL.	None.	None.
NWW1004	No method specified.	The specified URL is not correct.	Correct the URL.	None.	None.
NWW1005	ASCII Unicode conversion failed.	The transferred data has to be converted.	None.	None.	Contact Software AG.
NWW1006	Unicode ASCII conversion failed.	The transferred data has to be converted.	None.	None.	Contact Software AG.
NWW1007	Method ... not found.	A specified method cannot be called.	Correct the URL.	Add method to your class.	Check your registry configuration.
NWW1008	Class ... not found.	A specified class cannot be called.	Correct the URL.	Create class and register with REGISTER *	Check your registry configuration.
NWW1009	Initialization of Class ... failed.	A specified class cannot be called.	Correct the URL.	Create class and register with REGISTER *	Check your registry configuration.
NWW1010	DCOM call failed, error	The call to DCOM failed.	None.	None.	Check your configuration.
NWW1011	DCOM error ... occurred ...	Internal DCOM error. See DCOM documentation for further information.	Correct the URL.	Correct the program.	Correct the Configuration/Installation.
NWW1012	DCOM initalisation failed.	The initial call to DCOM failed.	None.	None.	Correct the Configuration/Installation.
NWW1013	DCOM release failed.	The deletion of class and close of DCOM failed.	None.	None.	Correct the Configuration/Installation.
NWW1036	DCOM Class Name too long.	Natural allows only library names up to 32 characters.	Check the URL.	Check the URL specification.	None.

NWW1037	DCOM Method Name too long.	Natural allows only subprogram names up to 32 characters.	Check the URL.	Check the URL specification.	None.
---------	----------------------------	---	----------------	------------------------------	-------