Natural für Großrechner

Web Technology

Version 4.2.6 für Großrechner

Februar 2010

Natural

Dieses Dokument gilt für Natural für Großrechner ab Version 4.2.6 für Großrechner.

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

Inhaltsverzeichnis

1 Web Technology

This documentation provides an overview of the Natural web technologies and a short summary of their functions.

The following topics are covered:

	Natural Web Interface	The Natural Web Interface is a link between a Web Server (HTTP server) and your Natural environment.
	XML Toolkit	The XML Toolkit enables developers to process XML documents within Natural.

2 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 Configuration**
- **Natural Web Interface Essentials**
- **Natural Web Online Documentation SYSWEB**

3 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

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. Apache or Microsoft 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.

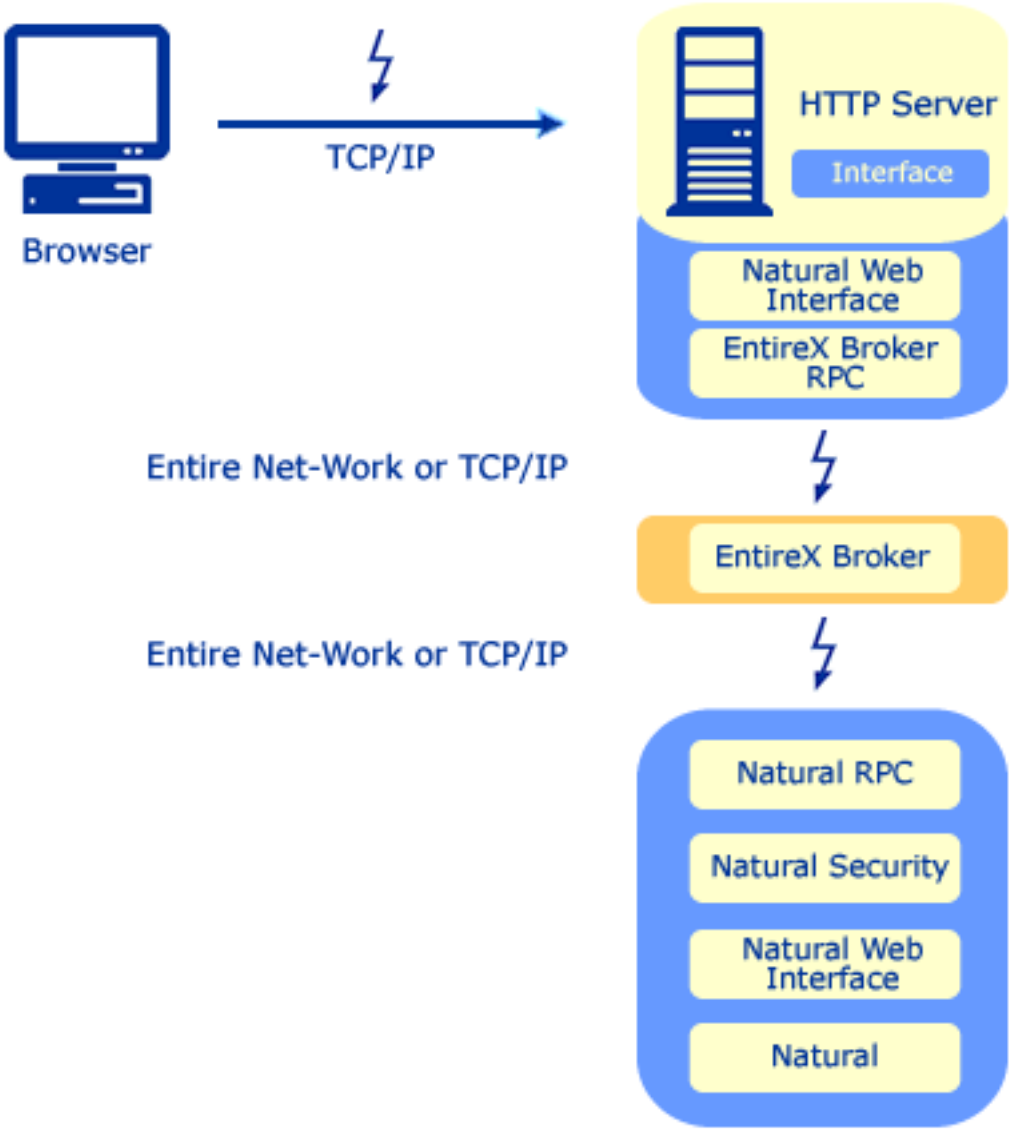
An interactive application enables users to input information and react by issuing output depending on that input. Examples of Web-based applications are order entry systems, travel booking services and parcel tracking systems. This considerably increases the scope of Natural applications. Not just in-house 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

The following topics are covered:

Communication Using Natural RPC 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 Communicator including EntireX Broker using RPC 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 contain 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 Entire System Server) 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.

On Natural for Windows, 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).

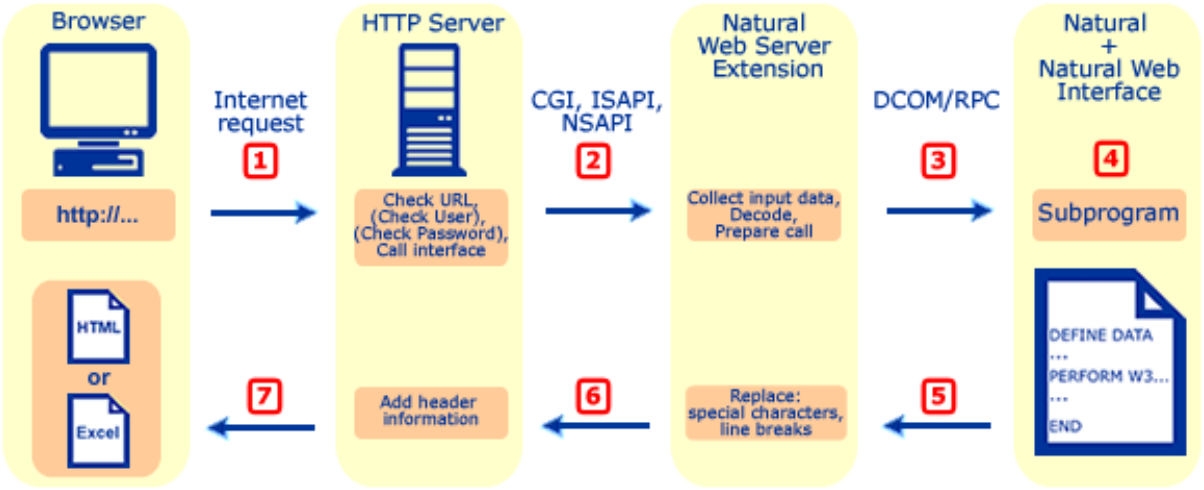
Web Page Creation

Web pages can be created with standard tools (e.g. Macromedia Dreamweaver or Microsoft Frontpage) or with the web page creation tool using the Natural generation functionality. From the Natural server, subprograms can be generated. There is no need to acquire knowledge about 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, PAL 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 a 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 authentification 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 when the initialization file is started, 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**.

4 Natural Web Interface Configuration

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.

5 Configuring the Natural Web Interface

This section provides information on how 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:

The latest documentation updates are published on Software AG's documentation site: **http://documentation.softwareag.com**.

Supported HTTP Servers

Operating System	HTTP Server
Windows (Intel)	■ Microsoft Internet Information Server Version 5.0/6.0 ■ Apache Version 2.0.x ■ Apache Version 2.2.x
(*)	■ Apache Version 2.0.x ■ Apache Version 2.2.x

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 *EntireX Communicator* documentation.

The following topics are documented below:

- Current Version of Natural for Mainframes, UNIX or Windows

- EntireX Communicator / EntireX Developer´s Kit

Current Version of Natural for Mainframes, UNIX or Windows

On Windows and UNIX Systems using SYSWEB

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

- In the *Natural Execution Configuration* parameter group, add the two steplibs SYSWEB and SYSEXT to the steplib parameter subsection.

On Windows and UNIX Systems using SYSWEB3

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

- In the *Natural Execution Configuration* parameter group, add the two steplibs SYSWEB3 and SYSEXT to the steplib parameter subsection.

In a Mainframe Environment using SYSWEB

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. If required, 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 Communicator / EntireX Developer´s Kit

On Windows 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 Web Interface

The following topics are covered below:

Natural Web Interface

For mainframe, Windows and UNIX environments no configuration is required.

Natural Web Server Extensions for RPC

Adjust the configuration file using an external editor:

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

With a Natural RPC Server Running in a non-ASCII Environment

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

Natural Web Server Extensions for DCOM

Local DCOM (All Platforms)

No adjustments are required 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 (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 (Apache)

If you use the Apache Server, the default settings for User/Group specified at httpd.conf can be used:

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

Configuring an HTTP Server

Windows (Internet Information Server 5.0 and 6.0)

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

Communication with Natural Security

The new version of the EntireX Developer´s Kit supports the usage of two passwords and user IDs.

The first user ID 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` and `NWW_PASSWORD` have to be specified.

Natural Security

A second User ID/Password (`RPC_USER_ID`, `RPC_PASSWORD`) has to be set.

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

It is necessary to set up 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 compatibility with the existing implementation.

6 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: ■ 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 the server extension can be started from the command prompt and does not run when called by the HTTP Server, check whether the .INI file can be found if it is copied to the same directory your HTTP server is started from.
NWW0011 ERX error 00000000 occurred. Severity = Success Message:... 9999 NAT0935 Conflicting number of parameters (Subprogram...). Lib=... Pgm=D3MENU.	Wrong supbrogram called, or wrong Steplib used	Check your Call: ■ Check if the called subprogram uses the parameter data area W3PARM. ■ Check if the RPC Server uses the Steplib SYSWEB if called from a nww* interface. ■ Check if the RPC Server uses the Steplib SYSWEB3 if called from a nww3* interface. ■ Check if the called program is compiled with the correct SYSWEB/SYSWEB3 Library - Call NAT-DIR (see docu) to see

Error	Description	Recommended Action
		what interface has been used during compile time.
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: ■ 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: ■ 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: ■ Does the program really exist? ■ If the program exists, check your logon library or the steplibs 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.
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 generates a new ETID for every running Natural.

7 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 Communicator which are relevant for the operation of the Natural Web Interface. For more information, see the EntireX Communicator 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 documention 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.
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.

8 Working with the Natural Web Interface

This section covers the following topics:

Setting up your Environment

Prerequisites on the Web Environment Side

The following software must be installed:

On the web client	Browser software, such as Mozilla Firefox or Microsoft Internet Explorer.
On the web server	HTTP server software, such as Apache Server or Microsoft Internet Information Server.

Middleware Prerequisites

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

RPC	The broker of the Software AG product EntireX Communicator must be installed (for installation information, see the EntireX Communicator documentation).

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

Prerequisites on Natural Server Side

For Natural Web Interface **SYSWEB** the following prerequisites must be met:

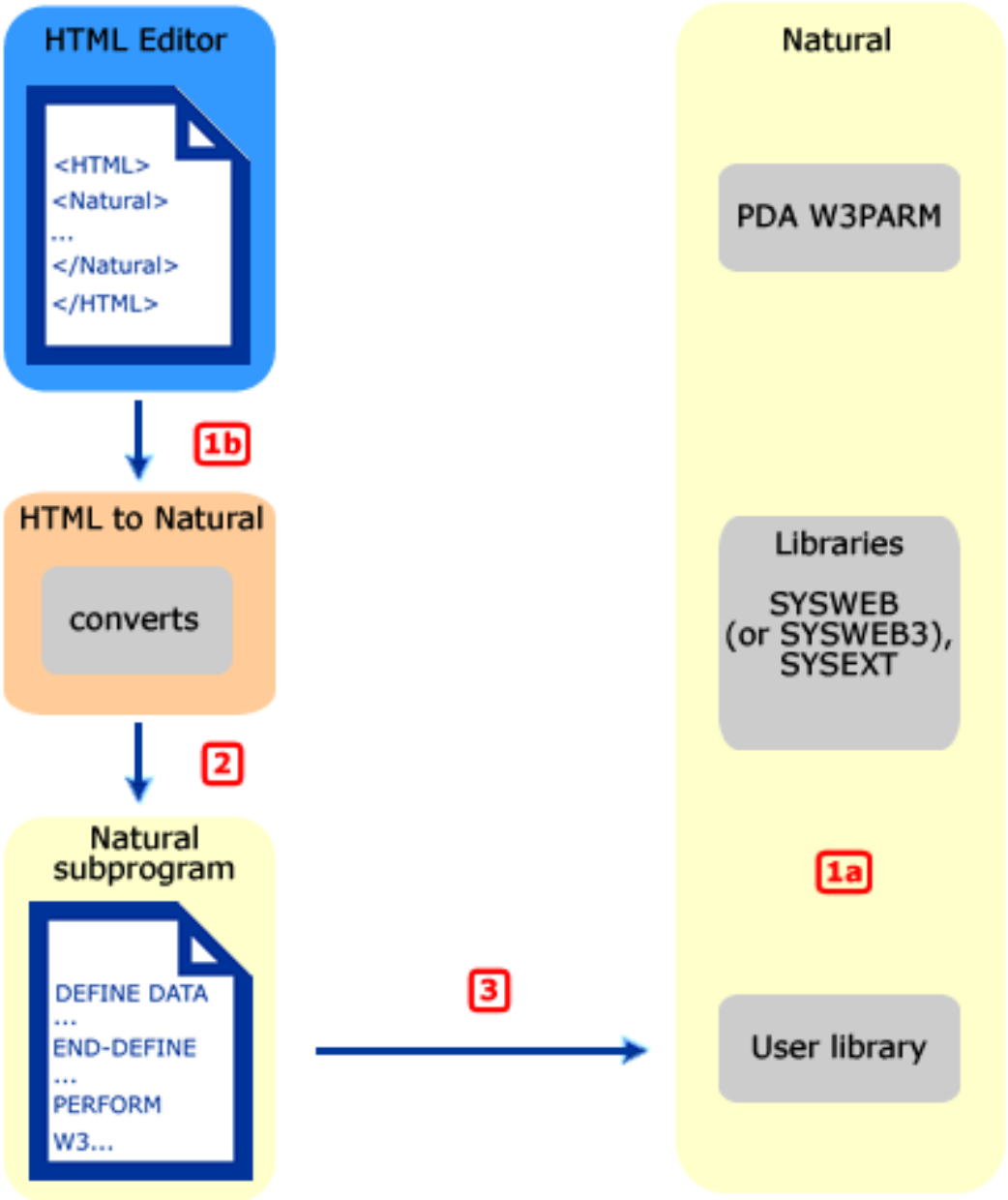
- Current Natural Version must be installed.
- The library SYSWEB.
  Either Natural steplibs must be available or the contents of the library SYSWEB must be copied to the library SYSTEM or to the user library that will be called by the RPC.
- The parameter data Area W3PARM.
- The Natural RPC stub or NaturalX.

Building Subprograms in Natural

The following diagram illustrates how you can build a subprogram:

1. Using an HTML editor
2. You use an 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.

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. You can either start coding directly in Natural or use an HTML editor.

Alternative 1: Coding Directly In Natural

When coding directly in Natural again 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 SYSWEB3 (or SYSWEB respectively); 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 W3PARM
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 W3PARM
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
```

Alternative 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 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
* ----- 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
*
* 22.09.03
*
--></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 W3PARM
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
*
* 22.09.2003
*
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`
: If you enter the `##HTTP_NEWLINE` string into your HTML, you can use all the subroutines beginning with `W3TEXT` in the library SYSWEB3 (or SYSWEB) to create a physical new line by compressing `#HTTP_NEWLINE` into the string by using `W3TextDynamic`.

`##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.

Testing Subprograms

There are three ways to test your subprograms:

When using SYSWEB:

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.

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.

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

Subprograms NAT*
: The `NAT*` subprograms are utilities that can be called from the Internet.

Natural Text Members T3*
: The `T3*` text members describe the contents of the library SYSWEB, 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*
: The `D3*` and `D4*` members are demonstration applications.

Programs Web*
: The `Web*` programs are utilities that run from the Natural NEXT prompt.

9 Natural Web Server Extensions

This document comprises the following sections:

Introduction for SYSWEB	Get an insight into the working and installation procedures of the Natural Web Server Extensions when using SYSWEB.
Initialization File	Here we describe parameters and variables of the initialization file.
Error Messages	Here we list likely errors.

10 Natural Web Server Extensions - Introduction for SYSWEB

This section covers the following topics:

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 call to the requested Natural program using a standard parameter data area. Calls are transmitted by the EntireX Broker that is included in EntireX Communicator.

As of Natural 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.
- Netscape Server Application Programming Interface (NSAPI) only for Netscape FastTrack Server.

Installation - RPC

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
CGI	nwwcgi nwwcgi.ini
ISAPI	nwwisapi.dll nwwisapi.ini
NSAPI	nwwnsapi.dll nww/nsapi
Parameters	RPC_ETB_ID_NAME = broker name RPC_SERVER_NAME = service name 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
CGI	http://*server-name*/*cgi-library*/nwwcgi/*your-library*/*your-program*
ISAPI	http://*server-name*/*cgi-library*/nwwisapi.dll/*your-library*/*your-program*
NSAPI	http://*server-name*/nww/nsapi/*your-library*/*your-program*

11 Natural Web Server Extensions - Initialization File

This section covers the following topics:

General Information

The Natural Web Server Extension processes 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 characters, so parameters can be specified multiple times.

RPC Parameters

These parameters are required 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.
RPC_USER_ID	User ID used for the RPC. If not specified, either ■ NWW_USER_ID is used or ■ REMOTE_USER is used, if REMOTE_USER is set to 1
RPC_PASSWORD	User password used for the RPC. If not specified NWW_PASSWORD is used.
RPC_SSL_PARAMETER	Connect string for RPC using SSL.

DCOM Parameters

This parameter is required for the communication with DCOM (on Windows platforms only).

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 defines 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 CGI string. Sample Log Entry: `[Thu Jun 28 10:51:19 2005] nwwcgi.exe 04.02.05 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 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_FORMAT NWW_INOUT_LENGTH	**Anmerkung:** Use these parameters with SYSWEB only. Defines the amount of the transferred data. These parameters define the dimension of the parameter Out_Page of the IDL file.

Parameter	Description

	Used IDL File:
	
	`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`

NWW_PASSWORD	Defines the password for the user ID.
NWW_PATH_INFO	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. Example: `NWW_PATH_INFO=/syshtml/nat-env`
NWW_PATHINFO_PREFIX	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 file name (e.g. */nww/my.nww*/sysweb/nat-env). This prefix (shown in *italics*) has to be removed. Use this parameter to remove the specified prefix. Example: `NWW_PATHINFO_PREFIX=/nww/my.nww`
NWW_OUT_CSS	Replaces the strings with the specific character(s): String Character -->| (Tab) @ @ [[\ \ ]] { { | / } } 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.
NWW_OUT_CSS_TRANSLATE	Replaces the specified characters with the corresponding hexadecimal values: (Default value for ASCII)

Parameter	Description
	Character Hexadecimal value -->| (Tab) 09 @ 40 [5B \ 5C] 5D { 7B | 7C } 7D Example for English EBCDIC (Code Page 37): #(tab), @, [, \,], {, |, } NWW_OUT_CSS_TRANSLATE=05,7C,AD,61,BD,C0,4F, D0
NWW_USER_ID	User ID used for the RPC.
NWW_RETRY	If an error (NAT3009 Transaction aborted) occurs, this parameter defines how often the program will be called again. Default is 3.
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 delivers 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 to 1 also, 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 that are 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 further information on variables, see **http://hoohoo.ncsa.illinois.edu/cgi/env.html**.

Additional Variables

With the Natural Web Server Extension, it is possible to transfer additional 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 2005	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.

Environment Variable	Description
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%" cellspacing="0" cellpadding="5">
 <TR bgcolor="#CCFFCC">
   <TD><H2 align="center">
        $NWW_RUN Error Report
     </H2>
     <P align="center">
    <I><SMALL>Natural Web Server Extension Interface: $NWW_VERSION</SMALL></I></TD>
 </TR>
 <TR>
   <TD><B>The following error has been logged in the error log file:</B></TD>
 </TR>
</TABLE>
<TABLE border="0" width="100%" cellspacing="15" cellpadding="0">
 <TR valign="top">
   <TD align="right"><B>$SCRIPT_NAME:</B></TD>
   <TD><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><PRE>$RPC_ERROR
</PRE>
   </TD>
 </TR>
 <TR valign="top">
   <TD align="right"><B>for:</B></TD>
   <TD><TT>$SERVER_NAME:$SERVER_PORT</TT></TD>
 </TR>
 <TR valign="top">
   <TD align="right"><B>path:</B></TD>
   <TD><TT>$PATH_INFO</TT></TD>
 </TR>
</TABLE>
<TABLE border="0" width="100%" cellspacing="0" cellpadding="5">
 <TR bgcolor="#CCFFCC">
   <TD>NWW Error Template - $NWW_LOGTIME</TD>
   <TD align="right">Natural</TD>
```

```
 </TR>
</TABLE>
<P>
$NWW_ENVIRONMENT
</BODY></HTML>
```

12 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
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 instal
NWW0011	ERX error ... occurred ...	Internal ERX error. See EntireX Communicator documentation for further information. If the error contains the following part: *Message:* *...* *Program = NATSRVD* *...* *Error = 00082* *...* The called program does not work.	Correct URL.	Check and stow your program.	Check instal
NWW0012	ERX error register.	EntireX RPC Service cannot be initialized.	None.	None.	Check confi

Error Number	Error Message	Description	User	Programmer	Administrator
NWW0013	erx.dll cannot be loaded. Subcode:	EntireX Communicator erx.dll not found.	None.	None.	Check installation.
NWW0014	ERX logon failed.	EntireX Communicator logon cannot be performed.	Check User-ID, Password	Check installation file for Check User-ID, Password.	Check installation.
NWW0015	ERX logoff failed.	Logoff from EntireX Communicator 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 for NSAPI.
NWW0034	NWW_USER_ID too long.	User ID only with a maximum of 8 characters allowed.	None.	None.	Specify only user IDs 8 characters or fewer if other system will a more.
NWW0035	NWW_PASSWORD too long.	Passwords only with a maximum of 8 characters allowed.	None.	None.	Specify only (user-) passwords with 8 cha or fewer, even if othe system will allow mo
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 than 30000.	The output returned to the HTTP server is limited to restrictions of Natural RPC.	None.	None.	Change configuratio
NWW0101	Number of parameters is greater than 200.	The input parameter given from the HTTP server is limited to 200 parameters.	None.	Reduce number of parameters transferred to the Web Interface.	None.
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.

Error Number	Error Message	Description	User	Programmer	Administrator
NWW0815	Interface A1(1:v) no longer supported.	Wrong interface specified.	None.	None.	Remove NWW_INO and NWW_INO parameter.
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 Soft
NWW1006	Unicode ASCII conversion failed.	The transferred data has to be converted.	None.	None.	Contact Soft
NWW1007	Method ... not found.	A specified method cannot be called.	Correct the URL.	Add method to your class.	Check your configuratio
NWW1008	Class ... not found.	A specified class cannot be called.	Correct the URL.	Create class and register with REGISTER *	Check your configuratio
NWW1009	Initialization of Class ... failed.	A specified class cannot be called.	Correct the URL.	Create class and register with REGISTER *	Check your configuratio
NWW1010	DCOM call failed, error	The call to DCOM failed.	None.	None.	Check your
NWW1011	DCOM error ... occurred ...	Internal DCOM error. See DCOM documentation for further information.	Correct the URL.	Correct the program.	Correct the Congigurati
NWW1012	DCOM initialization failed.	The initial call to DCOM failed.	None.	None.	Correct the Configuratio
NWW1013	DCOM release failed.	The deletion of class and close of DCOM failed.	None.	None.	Correct the Configuratio
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.

13 Programming Tips

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

This section covers the following topics:

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 vs. 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_NEWLINE
Writing to the return page, a physical new line can be created by compressing the string ##HTTP_NEWLINE into the string.

##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.

Creating a New Page

If your output may exceed the limits of your return page, use `W3COUNTER` to evaluate how may 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 Communicator 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.

14 Web Interface Administration

This section covers the following topics:

Set the Size of the Return-Page Transport Buffer

This section applies to SYSWEB only.

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 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 above).

2 Recatalog all W3* sources from the library SYSWEB.

3 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 Initialize ##HTTP_RETURN_PAGE_MAX in the local data area W3LIMITS.

This defines the maximum length of return page.

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.

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 W3-TEXT-TO-HTML will call W3-ASCII-HTML-TABLE.

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.

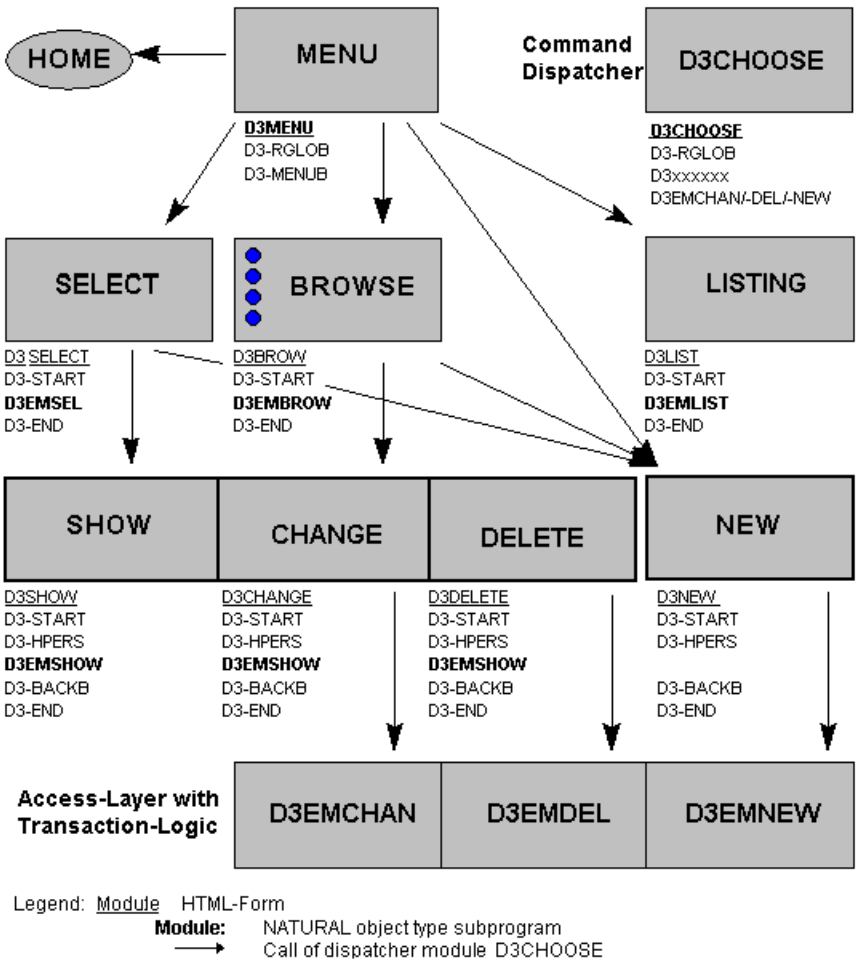
15 Demonstration Application - without JavaScript

This section covers the following topics:

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.



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 with SYSWEB:
`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 with SYSWEB:
`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 EMPLOYEES.

Perform a CATALL 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.

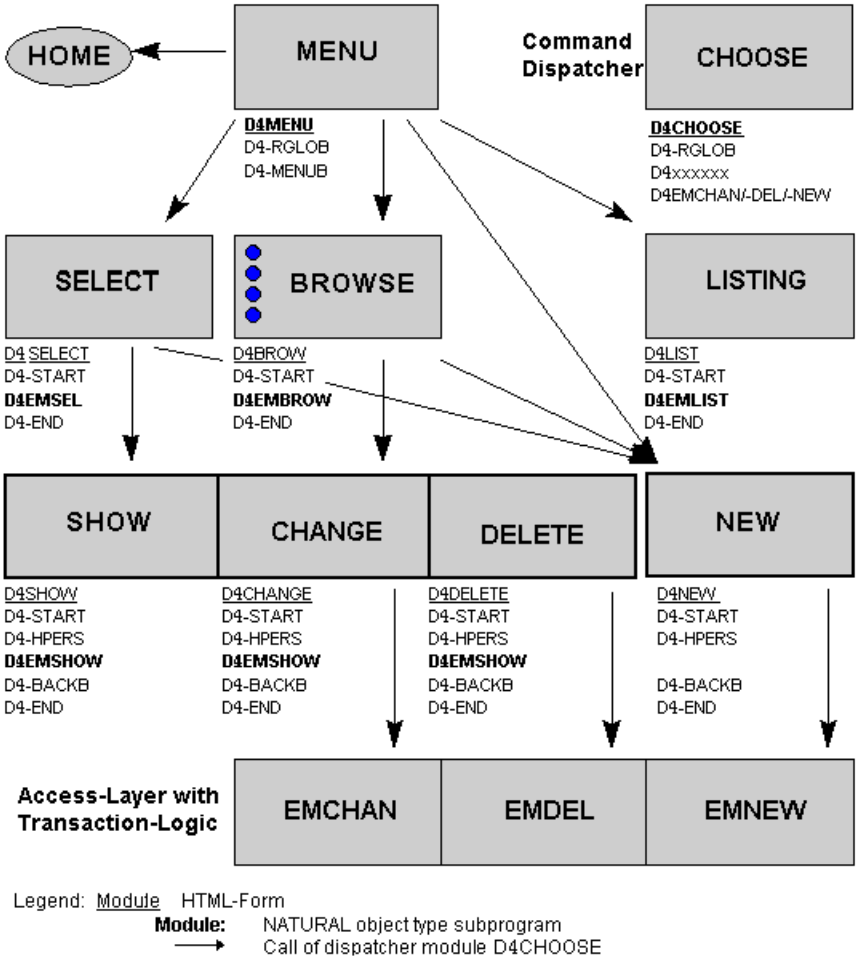
16 Demonstration Application - with JavaScript

This section covers the following topics:

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.



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 CATALL 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.

17 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.

18 Natural Web Online Documentation SYSWEB

This section covers the following topics:

General Information

The online documentation files are prefixed with E3* and T3*. The online documentation contains example programs that can be displayed and executed online. Depending on your installation of the Natural Web Interface, call the subprogram `NAT-DOCU` from the library `SYSWEB` to display the main page of online documentation at your web-browser.

Example of the URL to call the online documentation:
`http://yourserver/yourcgi/sysweb/nat-docu`

Note: To display the online documentation, the HTTP Server Extensions of the Natural Web Interface, must be installed, and a correct Natural RPC/DCOM Server has to be started. To access the program USR1057N, of library SYSEXT, add a steplib to SYSEXT or copy the programs to your system library.

Definition of Parameters	
`i/`	Input Variable
`o/`	Output Variable
`/o`	Optional Variable
`/m`	Mandatory Variable (has to be specified)
`/M`	Mandatory Variable. If empty, specific parts will not be generated.
`/H`	Variable will be translated to HTML
`/X`	Variable will be translated to XML
`/U`	Variable will be translated to URL

Basic Modules

The basic module names of the Natural Web Interface, start with the prefix `W3`.

They provide the communication between Natural Subprograms and the HTTP Server Extension. All other programs of the Natural Web Interface use these programs.

It is possible to make some administrative changes to define the amount and format of the transferred data, to change conversion tables and to change the error page.

Note: All new programs available are displayed in *italics* font in the tables below. Program names in brackets will be discontinued in the near future. Please use the program name mentioned in the description.

Program	Description
W3CLEAR	**Clear** the output page.
W3CONTENT-TYPE	Sets the **content type** of a document.
W3COUNTER	Returns the maximum number of bytes and the number of currently written or free bytes in the **output area**.
W3ERROR W3ERROR-TEMPLATE W3ERROR-TEMPLATE-XML W3ERROR-TEXT	Generates a default **error page.**
W3HTML W3HTMLLINE *W3HTMLDYNAMIC* *W3HTMLLINEDYNAMIC* W3HTMLARRAY	Writes an **HTML string** to the output page and converts special characters to an HTML-valid representation.
W3HTTP *W3HTTPDYNAMIC* W3HTTP-HEADER W3HTTPARRAY	Writes **HTTP settings** to the output page.
W3INFO	Returns **internal** settings.
W3INIT W3END	**Initializes** SYSWEB and prepares the document for returning to the HTTP server.
W3LIST-ENVIRONMENT W3LIST-ENVIRONMENT-TO-DYNAMIC	Lists all **variables**.
W3LOCATION	Sets the **location** of a page that is to be called instead this page.
W3READ-ENVIRONMENT W3READ-ENVIRONMENT-ARRAY *W3READ-ENVIRONMENT-TO-DYNAMIC*	Reads a variable **sent** by the HTTP server.
W3READ-ENVIRONMENT-TEXTAREA	Reads a variable set by a text area and splits the variable into **separate** lines.
W3READ-ENVIRONMENT-GROUP	Reads all environment variables with the **same** name.
W3TEXT W3TEXTLINE *W3TEXTDYNAMIC* *W3TEXTLINEDYNAMIC* W3TEXTARRAY	Writes a **text string** to the output page.
W3NEWLINE	Writes a **linebreak** to the output page.
(W3SPACE)	replace with -> `W3TEXTDYNAMIC " "`
W3-QUOTE-DQUOTE	Returns special character independent of characterset.
W3TEXT-TO-HTML *W3-ASCII-HTML-TABLE*	Converts ASCII to the specific encoding of **HTML.**
W3TEXT-TO-XML *W3-ASCII-XML-TABLE*	Converts ASCII to the specific encoding of **XML**.

Program	Description
W3TEXT-TO-URL *W3-ASCII-URL-TABLE*	Converts ASCII to the specific encoding of **URL**.

Output Post-Processing

Program	Description
W3REPLACE	Search the output page for a specific string and **replace** with a new one.
W3READ-OUTPUT	Read the **already written** output page.

HTML Extension

The prefix H3 is used for all program names of the HTML extension. This external subroutines, delivered with source code, generate HTML and use the basic modules of the Natural Web Interface.

The programs do not cover the complete syntax of HTML. They also do not support special enhancements of specific web browser. If you need enhancements, feel free to extend the programs delivered in source code, or create your own ones.

Program	Description	HTML Tag
H3-ANCHOR	Creates an **anchor** tag.	<A...>...</A>
H3-BUTTON	Creates reset/submit **buttons.**	<INPUT...>
H3-CHECKBOX-GROUP	Generates a **checkbox** group.	<INPUT...>
H3-COMMENT	Creates a **comment** line.	<!...>
H3-HEADER	Generates a **header** tag.	<Hn>
H3-IMAGE	Generates an **image** tag.	<IMG...>
H3-INPUT	Generates a text, password or hidden **input** field.	<INPUT...>
H3-LINE-BREAK H3-LINE_BREAK	Sets a **line break** with or without additional text.	
H3-OPEN-FORM H3-CLOSE-FORM	Starts a **form** tag for input fields.	<FORM>...</FORM>
H3-OPEN-HTML H3-OPEN-HTML-JAVASCRIPT H3-CLOSE-HTML	Starts and ends an **HTML** Document.	<HTML>...</HTML>
H3-OPEN-LIST H3-LIST-ITEM H3-CLOSE-LIST	Generates an ordered, unordered, menu or directory **list**.	<OL>...<LI>...</OL> <UL>...<LI>...</UL> <DIR>...<LI>...</DIR>

Program	Description	HTML Tag
		<MENU>...<LI>...</MENU>
H3-PARAGRAPH	Generates a **paragraph** with additional text.	<P...>
H3-RADIO-GROUP	Generates a **radio button** group.	<INPUT...>
H3-RULE	Sets a horizontal **rule**.	<HR...>
H3-SCROLLING-LIST	Generates a **scrolling** list.	<SELECT> ...<OPTION>... <SELECT>
H3-TABLE H3-TABLE-COLOR	Generates a **table**.	<TABLE > ... <TR> <TH>...</TH> </TR> <TR> <TD...>...</TD> </TR> ... </TABLE>
H3-TAG	Generates a universal **tag**.	<tag>
H3-TEXT-AREA	Generates a '**text area**'.	<TEXTAREA>...</TEXTAREA>
H3-TEXT-TO-HTML	Converts the content of a Natural string to '**HTML**'. replace with -> W3TEXT-TO-HTML	
H3-TEXT-TO-URL H3-ASCII-URL-TABLE	Converts the content of a Natural string to '**URL decoded**'. replace with -> W3TEXT-TO-URL	
H3-TIME_DATE H3-TIME-DATE	Generates a '**time/date**' string.	generated: Mon, 17 Jan 2005 15:35:18 GMT

Utilities

Online

Program	Description
WEB-WIZ	Generates the **skeleton** for a subprogram to be called by the NATURAL Web Interface.
WEB-ONL	Runs **Natural Web Interface** subprograms online.

Remote

Program	Description
NAT-LIB	Lists **all** Natural libraries.
NAT-DIR	Lists the **contents** of a specific Natural library.
NAT-ENV	Lists all parameters **passed** to a called Natural subprogram.
NAT-HTML	Displays a Natural **source** containing HTML.
NAT-INFO	Displays the current Natural Web Interface **settings.**
NAT-LIST	Displays a Natural **source object**.
NAT-DOCU	Displays the **online documentation**.

Demonstration Applications

The demonstration application delivered shows simple file maintenance with select functions. The demonstration is based on the Adabas file EMPLOYEES. To run the application, Adabas has to be active.

Two implementations of the demonstration applications are delivered:

1. one using JavaScript, name prefix D4*
   Depending on your installation of the HTTP Server Extensions, call the subprogram D4ENTER from the library SYSWEB.

   Example of the URL to call the demonstration application:
   `http://yourserver/yourcgi/sysweb/d4enter`
2. one using standard HTML 3.2, name prefix D3*
   Depending on your installation of the HTTP Server Extensions, call the subprogram D3MENU from the library SYSWEB.

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

All pictures used are delivered with the Natural Web Interface. Save them in the directory `pictures` on your HTTP-server in the remote directory `PICTURES`. If you want to use another remote directory name, set the environment variable `PICTURES` at the initialization file of your HTTP Server Extension with the specific remote directory name.

A JavaScript file for the D4* example is delivered with the Natural Web Interface. Save it in the directory `javascript` on your HTTP-server in the remote directory `/javascript`. If you want to use another remote directory name, set the environment variable `JAVASCRIPT` at the initialization file of your HTTP Server Extension with the specific remote directory name.

19 Clear Output Area

Subroutine Name	Executable Example	Viewable Example
W3CLEAR	E3CLEAR	E3CLEAR

Description

Deletes all data already written to the output area.

Parameters

```
*/ NONE
```

How To Invoke

```
PERFORM W3CLEAR
```

20 Set Document Content-Type

Subroutine Name	Executable Example	Viewable Example
W3CONTENT-TYPE	E3CONTYP	E3CONTYP

Description

Sets the content type of the document. This setting is used by the browser programs to find out how the content is to be displayed.

`W3CONTENT-TYPE` or `W3LOCATION` has to be the first output of a document.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.
There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Parameters

```
1 H3VALUE (A) DYNAMIC /* i /m : Content type to be set
```

How To Invoke

```
PERFORM W3CONTENT-TYPE H3VALUE
```

21 Count Size of Output Area

Subroutine Name	Executable Example	Viewable Example
W3COUNT	E3COUNT	E3COUNT

Description

Returns the current size of the output area and the number of bytes already written there.

Parameters

```
1 W3WRITTEN (I4) /* o/m : Currently written bytes
1 W3MAXPAGE (I4) /* o/m : Maximum bytes possible
1 W3FREE    (I4) /* o/m : Free bytes
```

How To Invoke

```
PERFORM W3COUNTER W3WRITTEN W3MAXPAGE W3FREE
```

22 Generate Error Page

Subroutine Name	Executable Example	Viewable Example
W3ERROR W3ERROR-TEMPLATE W3ERROR-TEMPLATE-XML W3ERROR-TEXT	E3ERROR	E3ERROR

Description

Errors generated by the Natural runtime should be handled to avoid screen output. Therefore, an `ON ERROR` section must be added to all programs called with the Natural Web Interface. The PDA `W3CONST` must be added as well.

The subroutine W3ERROR-TEMPLATE is called if an error occurs. This routine can be changed for your own needs.

The subroutine W3ERROR-TEMPLATE-XML returns the error page as XHTML page. This routine can be changed for your own needs. To activate this template, uncatalog W3ERROR-TEMPLATE and rename the subroutine from W3ERROR-TEMPLATE-XML to W3ERROR-TEMPLATE and stow.

The subroutine W3ERROR-TEXT is for internal use only.

Parameters

```
1 ##W3ERROR
  2 NR         (I4)   /* i /m : Number of the error
  2 LINE       (I4)   /* i /m : Line in the Natural program
  2 SUBPROGRAM (A008) /* i /m : Subprogram name
  2 SUBROUTINE (A032) /* i /m :  Subroutine name
  2 TEXT       (A250) /* i /m : Error text
```

How To Invoke

```
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
```

23 Writes to the Document and Converts to Valid HTML

Subroutine Name	Executable Example	Viewable Example
W3HTML W3HTMLDYNAMIC W3HTMLLINE W3HTMLINEDYNAMIC W3HTMLARRAY	E3HTMLA	E3HTMLA

Description

Writes a string to the document and converts special characters, such as "<", ">", "Ã¼", ...

If you want to create a line break after your output, use `W3HTMLLINE` or `W3HTMLLINEDYNAMIC`. If you want to create a line break inside your string, compress ##HTTP-NEWLINE into your string.

W3HTML and W3HTMLLINE will delete trailing blanks from the given string.

For better performance use dynamic variables.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.
There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Parameters

1. W3HTML

```
1 H3HTML          (A) DYNAMIC BY VALUE /* i /mH: Output string
```

2. W3HTMLDYNAMIC

```
1 H3DYNAMIC       (A) DYNAMIC BY VALUE /* i /mH: Output string
```

3. W3HTMLLINE

```
1 H3HTML          (A) DYNAMIC BY VALUE /* i /mH: Output string
```

4. W3HTMLLINEDYNAMIC

```
1 H3DYNAMIC       (A) DYNAMIC BY       /* i /mH: Output string
```

5. W3HTMLARRAY

```
1 H3ARRAYVALUE  (A/1:v) DYNAMIC        /* i /mH: Output array
1 H3VALUELENGTH (I4)                   /* i /m : Length of output array
```

How To Invoke

```
PERFORM W3HTML H3HTML
PERFORM W3HTMLDYNAMIC H3DYNAMIC
PERFORM W3HTMLLINE H3HTML PERFORM W3HTMLLINEDYNAMIC H3DYNAMIC
PERFORM W3HTMLARRAY H3ARRAYVALUE H3VALUELENGTH
```

24 Writes HTTP Settings to the Document

Subroutine Name	Executable Example	Viewable Example
W3HTTP W3HTTPDYNAMIC W3HTTP-HEADER W3HTTPARRAY	E3HTTP	E3HTTP

Description

Writes a text line to the HEAD of a document. In these text line settings, you can specify COOKIES, EXPIRE-DATES or other settings of an HTTP-compatible document.

Physical new lines in the output can be created by compressing ##HTTP_NEWLINE into a Natural string.

If you want to create a line break inside your string, compress ##HTTP-NEWLINE into your string.

W3HTTP will delete trailing blanks from the given string.

For better performance use dynamic variables.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.
There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Parameters

1. W3HTTP

```
1 W3STRING      (A) DYNAMIC     /* i /m : Header name value pairs
```

2. W3HTTPDYNAMIC

```
1 W3DYNAMIC     (A) DYNAMIC     /* i /m : Header name value pairs
```

3. W3HTTP-HEADER

```
1 W3HTTPNAME    (A) DYNAMIC     /* i /m : Header name
1 W3HTTPVALUE   (A) DYNAMIC     /* i /m : Header value
```

4. W3HTTPARRAY

```
1 W3ARRAYVALUE  (A/1:V) DYNAMIC /* i /m : Header name value pairs
1 W3VALUELENGTH (I4)            /* i /m : Length of output array
```

How To Invoke

```
PERFORM W3HTTP W3STRING
PERFORM W3HTTPDYNAMIC W3DYNAMIC
PERFORM W3HTTP-HEADER W3HTTPNAME W3HTTPVALUE
PERFORM W3HTTPARRAY W3ARRAYVALUE W3VALUELENGTH
```

25 Info About Internal Values

Subroutine Name	Executable Example	Viewable Example
W3INFO	E3INFO	E3INFO

Description

This program enables you to set and read some internal values of the Web Interface.

Read (Action ' ')

The values for ERROR-NR VERSION, LOG-TIME, FORM, LIST(*) and LIST_MAX will be returned.

Set List (Action 'L')

For H3-OPEN-LIST, H3-CLOSE-LIST and H3-LIST-ITEM, an internal array is used to save the style of the generated list. This style will be used to generate the correct close tag.

Set Form (Action 'F')

For all programs, you can generate tags that can only be used inside a FORM tag. A flag can be called to check if a FORM is open or not. The flag will be changed by H3-OPEN-FORM and H3-CLOSE-FORM.

Parameters

```
LOCAL USING WPINFO
```

PDA W3PINFO

```
1 ##W3INFO
  2 ACTION   (A1)        /* i /m : Action to be called
  2 LOG-TIME (A030)      /*  o/m : Log time set by the Natural Web Interface
  2 VERSION  (A015)      /*  o/m : Version set by the HTTP Server Extension
  2 WEBAPI   (A015)      /*  o/m : Version set by the Natural Web Interface
  2 ERROR-NR (I4)       /*  o/m : Error number set by the Natural Web Interface
  2 FORM     ( L)        /* io/m : Indicates whether a FORM is open
  2 LIST     (A001/1:10) /* io/m : Saves the type of LIST
  2 LIST_MAX (N002)      /* io/m : Current number of nested LISTs
```

How To Invoke

```
PERFORM W3INFO ##W3INFO
```

26 End and Initialize Document

Subroutine Name	Executable Example	Viewable Example
W3INIT W3END	E3END	E3END

Description

Each Program needs to initialize and end the web interface by special programs. The initialisation is done by `W3INIT`.

The W3PARM PDA must be passed to initialize passed parameters for further use.

`W3END` ends the document and prepares the return to the HTTPserver. The `W3PARM` PDA defined at the initial program has to be passed to `W3END`.

If `W3*` calls are performed after `W3END`, the written output will not be transferred to the HTTP server.

Parameters

1. W3INIT

```
USING W3PARM /* io/m : Parameter of Subprogram
```

2. W3END

```
USING W3PARM /* io/m : Parameter of Subprogram
```

3. W3PARM

```
1 ##RPC
  2 VERSION           (A010)     /* i /m : Interface version
  2 LOG-TIME          (A030)     /* i /m : Timestamp
  2 RETURN_PAGE       (A250/1:V) /* io/m : Transfer area
  2 RETURN_PAGE_COUNT (I004)     /* io/m : Bytes sent
  2 ERROR-NR          (I004)     /*  o/m : Error number
```

How To Invoke

```
PERFORM W3INIT ##RPC
PERFORM W3END ##RPC
```

27 List All Environment Variables

Subroutine Name	Executable Example	Viewable Example
W3LIST-ENVIRONMENT W3LIST-ENVIRONMENT-TO-DYNAMIC	E3ENVLIS	E3ENVLIS

Description

List all variables sent by the HTTP server.

Parameters

1. W3LIST-ENVIRONMENT

```
1 W3START            (I4)       /* io/m : Offset to be started at
*                               /*        out: 0 no occurrences

*                               /*        out: >0 more occurrences
1 W3ARRAYCOUNTER     (A72/1:V)  /* io/m : Length of array returned values
1 W3ARRAYNAME        (A250/1:V) /*  o/m : Name of variables
1 W3ARRAYVALUE       (I4/1:V)   /*  o/m : Value of variables
1 W3ARRAYVALUELENGTH (L/1:V)    /*  o/m : Length of variables
1 W3ARRAYVALUESERVER (I4)       /*  o/m : Variable belongs to
1 W3ARRAYMAXIMUM                /*  o/m : Total number of variables
```

2. W3LIST-ENVIRONMENT-TO-DYNAMIC

```
1 W3START            (I4)           /* io/m : Offset to be started at
*                                   /*        out: 0 no occurrences
*                                   /*        out: >0 more occurrences
1 W3ARRAYCOUNTER     (A/1:V)dynamic /* io/m : Length of array returned values
1 W3ARRAYNAME        (A/1:V)dynamic /*  o/m : Name of variables
1 W3ARRAYVALUE       (I4/1:V)       /*  o/m : Value of variables
1 W3ARRAYVALUELENGTH (L/1:V)        /*  o/m : Length of variables
```

```
1 W3ARRAYVALUESERVER (I4)           /*  o/m : Variable belongs to
1 W3ARRAYMAXIMUM                    /*  o/m : Total number of variables
```

How To Invoke

```
PERFORM W3LIST-ENVIRONMENT W3START W3NAME W3ARRAYCOUNTER
        W3ARRAYNAME(*) W3ARRAYVALUE(*) W3ARRAYVALUELENGTH(*)
        W3ARRAYVALUESERVER(*) W3ARRAYMAXIMUM
```

28 Set Document Location

Subroutine Name	Executable Example	Viewable Example
W3LOCATION	E3LOCAT	E3LOCAT

Description

Sets the location of a document that is to be loaded. This subroutine can be used to call a static page instead of a dynamic one from a Natural program.

`W3LOCATION` or `W3CONTENT-TYPE` has to be the first output of a document.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.
There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Parameters

```
1 W3CONTENT (A) DYNAMIC /* i /m : Content type to be set
```

How To Invoke

```
PERFORM W3CONTENT-TYPE W3CONTENT
```

29 Read Environment Variable

Subroutine Name	Executable Example	Viewable Example
W3READ-ENVIRONMENT-ARRAY W3READ-ENVIRONMENT W3READ-ENVIRONMENT-TO-DYNAMIC	E3ENVARY	E3ENVARY

Description

Reads the first occurrence of a single variable. With `W3READ-ENVIRONMENT-ARRAY`, a variable can be read beginning with an offset. This can be used to read multiple occurrences of the same variable.

Parameters

1. W3READ-ENVIRONMENT-ARRAY

```
1 W3START        (I4)       /* io/m : Offset to be started at
*                           /*        out: 0 no occurrences
*                           /*        out: >0 more occurrences
1 W3NAME         (A072)     /* i /m : Name of the variable to
*                           /*        be searched for
1 W3VALUESERVER  (A1)       /* i /m : Search for variables from
*                           /*        'S' server
*                           /*        'P' page or URL
*                           /*        ' ' both
1 W3ARRAYCOUNTER (I4)       /* io/m : Size of array,
                            /*        characters read
1 W3VALUEARRAY   (A001/1:V) /*  o/m : Array with the returned value
1 W3VALUELENGTH  (I4)       /*  o/m : length of the value
```

2. W3READ-ENVIRONMENT

```
1 W3NAME        (A072) /* i /m : Name of the variable
                       /*        searched for
1 W3VALUESERVER (A1)   /* i /m : Search for variables in
*                      /*        'S' server
*                      /*        'P' page or URL
*                      /*        ' ' both
1 W3VALUE       (A250) /*  o/m : Returned value
1 W3VALUELENGTH (I4)   /*  o/m : Length of the value
```

3. W3READ-ENVIRONMENT-TO-DYNAMIC

```
1 W3NAME         (A072)      /* i /m : Name of the variable
                             /*        searched for
1 W3VALUESERVER  (A1)        /* i /m : Search for variables in
*                            /*        'S' server
*                            /*        'P' page or URL
*                            /*        ' ' both
1 W3VALUEDYNAMIC (A) DYNAMIC /* o/m : Returned value
```

How To Invoke

```
PERFORM W3READ-ENVIRONMENT-ARRAY W3START W3NAME
        W3VALUESERVER W3ARRAYCOUNTER
        W3VALUEARRAY(*) W3VALUELENGTH

PERFORM W3READ-ENVIRONMENT W3NAME W3VALUESERVER W3VALUE
        W3VALUELENGTH

PERFORM W3READ-ENVIRONMENT-TO-DYNAMIC W3NAME W3VALUESERVER
        W3VALUEDYNAMIC
```

30 Read Environment Variables Groups

Subroutine Name	Executable Example	Viewable Example
W3READ-ENVIRONMENT-GROUP	E3ENVGRO?test=a&test=bb&test=cc	E3ENVGRO

Description

Reads all variables with the same name, e.g. set from a multiple select.

Parameters

```
1 W3START        (I4)        /* io/m : Offset to be started at
*                            /*        out: 0 no occurrences
*                            /*        out: >0 more occurrences
1 W3NAME         (A) DYNAMIC /* i /m : Name of variable
1 W3VALUESERVER  (A1)        /* i /m : Search for variables in
*                            /*        'S' server
*                            /*        'P' page or URL
*                            /*        ' ' both
1 W3ARRAYCOUNTER (I4)        /* io/m : Length of array,
                             /*        returned values
1 W3ARRAYVALUES  (A250/1:V)  /*  o/m : Values of variable
1 W3ARRAYMAXIMUM (I4)        /*  o/m : Total number of variables
```

How To Invoke

```
PERFORM W3READ-ENVIRONMENT-GROUP W3START W3NAME
        W3VALUESERVER W3ARRAYCOUNTER
        W3ARRAYVALUES(*) W3ARRAYMAXIMUM
```

31 Read Environment Text Area Variables

Subroutine Name	Executable Example	Viewable Example
W3READ-ENVIRONMENT-TEXTAREA	E3ENVTX	E3ENVTX E3ENVTX1

Description

Reads a variable set by a text area tag and separates the text lines.

Parameters

```
1 W3START            (I4)       /* io/m : Offset to be started at
*                               /*        out: 0 no occurrences
*                               /*        out: >0 more occurrences
1 W3NAME             (A072)     /* i /m : Name of variable
1 W3VALUESERVER      (A1)       /* i /m : Search for variables in
*                               /*        'S' server
*                               /*        'P' page or URL
*                               /*        ' ' both
1 W3ARRAYCOUNTER     (A250/1:V) /* io/m : Length of array,
                                /*        returned values
1 W3ARRAYVALUE       (I4/1:V)   /*  o/m : Value of variables
1 W3ARRAYVALUELENGTH (I4)       /*  o/m : Length of variables
1 W3ARRAYMAXIMUM                /*  o/m : Total number of variables
```

How To Invoke

```
PERFORM W3READ-ENVIRONMENT-TEXTAREA W3START W3NAME
        W3VALUESERVER W3ARRAYCOUNTER(*)
        W3ARRAYVALUE(*) W3ARRAYVALUELENGTH(*)
        W3ARRAYMAXIMUM
```

32 Write Text to Document

Subroutine Name	Executable Example	Viewable Example
W3TEXT W3TEXTDYNAMIC W3TEXTLINE W3TEXTLINEDYNAMIC W3TEXTARRAY	E3TEXT	E3TEXT

Description

Writes a character string to the document.

If you want to create a line break after your output, use `W3HTMLLINE` or `W3HTMLLINEDYNAMIC`.

If you want to create a line break inside your string, compress ##HTTP-NEWLINE into your string.

W3TEXT and W3TEXTLINE will delete trailing blanks from the given string.

For better performance use dynamic variables.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.
There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Parameters

1. W3TEXT

```
1 W3TEXT        (A) DYNAMIC     /* i /m : Output string
```

2. W3TEXTDYNAMIC

```
1 W3DYNAMIC     (A) DYNAMIC     /* i /m : Output array
```

3. W3TEXTLINE

```
1 W3TEXT        (A) DYNAMIC     /* i /m : Output string with new line
```

4. W3TEXTLINEDYNAMIC

```
1 W3DYNAMIC     (A) DYNAMIC     /* i /m : Output string with new line
```

5. W3TEXTARRAY

```
1 H3ARRAYVALUE  (A/1:v) DYNAMIC /* i /m : Output array
1 H3VALUELENGTH (I4)            /* i /m : Length of output array
```

How To Invoke

```
PERFORM W3TEXT W3TEXT
PERFORM W3TEXTDYNAMIC W3TEXT
PERFORM W3TEXTLINE W3TEXT
PERFORM W3TEXTLINEDYNAMIC W3TEXT
PERFORM W3TEXTARRAY W3ARRAYVALUE W3VALUELENGTH
```

33 Write Newline to Output Area

Subroutine Name
W3NEWLINE

Description

Adds a single newline (##HTTP_NEWLINE) to the output area.

This subroutine will be deleted in one of the next versions. Use W3TEXTDYNAMIC ##HTTP_NEWLINE instead.

Parameters

```
*/ NONE
```

How To Invoke

```
PERFORM W3NEWLINE
```

34 Text to HTML

Subprogram Name	Executable Example	Viewable Example
W3TEXT-TO-HTML W3-ASCII-HTML-TABLE	E3TX2HTM	E3TX2HTM

Description

Converts a string to HTML syntax. Useful if special characters are included.

The subprogram W3-ASCII-HTML-TABLE will be called from W3TEXT-TO-HTML and W3HTML and contains a list of all conversations that will be made.
This program can be changed and extended for the user's needs.

Parameters

```
1 W3HTML (A) DYNAMIC /* io/mH: HTML text conversion
```

How To Invoke

```
PERFORM W3TEXT-TO-HTML W3HTML
```

35 Text to XML

Subprogram Name	Executable Example	Viewable Example
W3TEXT-TO-XML W3-ASCII-XML-TABLE	E3TX2XML	E3TX2XML

Description

Converts a string to XML syntax. Useful if special characters are included.

The subprogram W3-ASCII-XML-TABLE will be called from W3TEXT-TO-XML and contains a list of all conversations that will be made.
This program can be changed and extended for the user's needs.

Parameters

```
1 W3XML (A) DYNAMIC /* io/mX: XML text conversion
```

How To Invoke

```
PERFORM W3TEXT-TO-XML W3XML
```

36 Text to URL

Subprogram Name	Executable Example	Viewable Example
W3TEXT-TO-URL W3-ASCII-URL-TABLE	E3TX2URL	E3TX2URL

Description

Converts a string to URL syntax. Useful if special characters are included.

The subprogram W3-ASCII-URL-TABLE will be called from W3TEXT-TO-URL and contains a list of all conversations that will be made.
This program can be changed and extended for the user's needs.

Parameters

```
1 W3URL (A) DYNAMIC /* io/mU: URL text conversion
```

How To Invoke

```
PERFORM W3TEXT-TO-URL W3URL
```

37 Replace Inside Return Document

Subroutine Name	Executable Example	Viewable Example
W3REPLACE	E3REPLAC	E3REPLAC

Description

Search the already written output page for a specific string and replace all occurrences with a new string. With the encoding parameter, the given data will be encoded before the replacement is done:

- " " -> no encoding
- "HTML" -> HTML encoding (e.g. < -> <)
- "URL" -> URL encoding
- "XML" -> XML encoding (e.g. < -> <)

Parameters

```
1 W3ENCODING (A) DYNAMIC BY VALUE /* i /m : encoding
1 W3OLD      (A) DYNAMIC BY VALUE /* i /m : old string
1 W3NEW      (A) DYNAMIC BY VALUE /* i /m : new string
```

How To Invoke

```
PERFORM W3REPLACE "$weather$" "fine, no clouds"
```

38 Read Output Page

Subroutine Name	Executable Example	Viewable Example
W3READ-OUTPUT	E3ROUT	E3ROUT

Description

Read into dynamic variables the output page already written.

Parameters

```
1 W3HEADER (A) DYNAMIC /* o/m : written header
1 W3BODY   (A) DYNAMIC /* o/m : written body
```

How To Invoke

```
PERFORM W3READ-OUTPUT W3HEADER W3BODY
```

39 Anchor

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-ANCHOR	H3ANCHOR	E3ANCHOR	E3ANCHOR

Description

Creates a hyperlink.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

`<A HREF="URL"> </A>`

Supported Attributes

`NAME=string`

Parameters

```
1 H3URL             (A) DYNAMIC /* i /m : URL of the Link. Enter
*                               /*        'THIS' to reference
*                               /*        the current page as URL.
1 H3NAME            (A) DYNAMIC /* i /M : Name of the anchor.
1 H3STRING          (A) DYNAMIC /* i /MH: String to be displayed
*                               /*        as anchor text.
```

How To Invoke

```
PERFORM H3-ANCHOR H3URL H3NAME H3STRING
```

40 Button

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-BUTTON	H3BUTTON	E3BUTTON	E3BUTTON

Description

Creates a reset/submit button.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

```
<INPUT TYPE="submit|reset|image" NAME="string"> </INPUT>
```

Supported Attributes

```
VALUE="string", SRC="URL"
```

Parameters

```
1 H3TYPE  (A1)        /* i /m : 'R' reset button
*                     /* 'S' submit button
*                     /* 'I' submit button with image
1 H3NAME  (A) DYNAMIC /* i /M : Name of the button
1 H3VALUE (A) DYNAMIC /* i /M : Value of the input variable
1 H3URL   (A) DYNAMIC /* i /M : URL of the picture to be used
```

How To Invoke

```
PERFORM H3-BUTTON H3TYPE H3NAME H3VALUE H3URL
```

41 Checkbox Group

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-CHECKBOX-GROUP	H3RCGROU	E3RCGROU	E3RCGROU

Description

Creates a checkbox group. The group can be formatted inside a table.

Appearance

```
<INPUT TYPE="CHECKBOX" NAME="string">
```

Supported Attributes

```
VALUE=string, CHECKED
```

Parameters

```
1 H3ARRAYCOUNT   (I4)            /* i /m : Number of group elements
1 H3ARRAYNAME    (A/1:V) dynamic /* i /m : Name of the group variable
1 H3ARRAYVALUE   (A/1:V) dynamic /* i /M : Default value of the
*                                /*        group variable
1 H3ARRAYLABEL   (A/1:V)         /* i /MH: Label of the group element
1 H3ARRAYCHECKED ( L/1:V)        /* i /M : Button selected by default
1 H3LINEBREAK    ( L)            /* i /m : Set line breaks between by
*                                /*        the elements
1 H3ROW          (N4)            /* i /m : Set number of rows for
*                                /*        tables
1 H3COLUMN       (N4)            /* i /m : Set number of columns for
*                                /*        tables
```

How To Invoke

```
PERFORM H3-CHECKBOX-GROUP H3ARRAYCOUNT H3ARRAYNAME(*)
        H3ARRAYVALUE(*) H3ARRAYLABEL(*)
        H3ARRAYCHECKED(*) H3LINEBREAK H3ROW H3COLUMN
```

42 Comment Line

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-COMMENT	H3COMMEN	E3COMMEN	E3COMMEN

Description

Creates a comment line inside an HTML page.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

```
<!-- value -->
```

Parameters

```
1 H3VALUE (A) DYNAMIC /* i /m : Value to set as comment
```

How To Invoke

```
PERFORM H3-COMMENT H3VALUE
```

43 Level n Header

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-HEADER	H3HEADER	E3HEADER	E3HEADER

Description

Creates a header of a specified level. Levels 1 to 6 are allowed.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

`<H1> </H1>` or

`<H2> </H2>` or

`<H3> </H3>` or

`<H4> </H4>` or

`<H5> </H5>` or

`<H6> </H6>`

Parameters

```
1 H3LEVEL (N2)        /* i /m : Level of the header
1 H3HTML  (A) DYNAMIC /* i /mH: HTML text to be set
```

How To Invoke

```
PERFORM H3-HEADER H3LEVEL H3HTML
```

44 Image

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-IMAGE	H3IMAGE	E3IMAGE	E3IMAGE

Description

Displays an image. The image itself cannot be saved inside Natural. Therefore, all pictures must be saved with the HTTP Server.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

`<IMG SRC="URL">`

Supported Attributes

`ALT=string, HEIGHT=n, WIDTH=n, ALIGN=left|right|top|middle|bottom`

Parameters

```
1 H3URL    (A) DYNAMIC /* i /m : URL of the picture source
1 H3STRING (A) DYNAMIC /* i /M : Alternative name string
*                      /*        for non-GUI browsers
1 H3HEIGHT (N4)        /* i /M : Height if the picture
1 H3WIDTH  (N4)        /* i /M : Width of the picture
1 H3ALIGN  (A1)        /* i /M : Align the picture to
*                      /*        'L' Left
*                      /*        'R' Right
*                      /*        'T' Top
*                      /*        'B' Bottom
*                      /*        'M' Middle
```

How To Invoke

```
PERFORM H3-IMAGE H3URL H3STRING H3HEIGHT H3WIDTH H3ALIGN
```

45 Input

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-INPUT	H3INPUT	E3INPUT	E3INPUT

Description

Creates an input field. Possible field types are text, password and hidden.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

```
<INPUT TYPE="text|password|hidden" NAME="string">
```

Supported Attributes

```
VALUE=string, MAXLENGTH=n, SIZE=n
```

Parameters

```
1 H3TYPE  (A1)        /* i /m : Type of the input field
*                     /*        'T' Text (default)
*                     /*        'P' Password
*                     /*        'H' Hidden
1 H3NAME  (A) DYNAMIC /* i /M : Name of the input variable
1 H3VALUE (A) DYNAMIC /* i /M : Default value of the input variable
1 H3SIZE  (N4)        /* i /M : Size of the input box
1 H3MAX   (N4)        /* i /M : Maximum length of the input text
```

How To Invoke

```
PERFORM H3-INPUT H3TYPE H3NAME H3VALUE H3SIZE H3MAX
```

46 Line Break

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-LINE-BREAK H3-LINE_BREAK	H3LBREA H3LBREAK	E3LBREAK	E3LBREAK

Description

Forces a line break, with or without additional HTML text.

Changes from previous versions

Both versions can be used equivalently, because the parameter is marked as optional. Use only the version `H3-LINE-BREAK`. H3-LINE-BREAK will be removed in one of the next versions.

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE. There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

`<BR>`

Parameters

1. H3-LINE-BREAK

```
1 H3HTML (A) DYNAMIC /* i /OH: HTML text after the line break
```

2. H3-LINE_BREAK

```
1 H3HTML (A) DYNAMIC /* i /OH: HTML text after the line break
```

How To Invoke

```
PERFORM H3-LINE-BREAK
PERFORM H3-LINE-BREAK 1X
PERFORM H3-LINE-BREAK H3HTML
```

47 Form

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-OPEN-FORM H3-CLOSE-FORM	H3OFORM H3CFORM	E3FORM	E3FORM

Description

Creates a form. You must perform `H3-OPEN-FORM` before and `H3-CLOSE-FORM` afterwards.

If no `H3-CLOSE-FORM` is performed, `H3-CLOSE-HTML` will close all open forms.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

`<FORM ACTION="`*`URL`*`" METHOD="`*`get|post`*`"> </FORM>`

Supported Attributes

`ACTION=`*`URL`*`, METHOD=`*`get|post`*

Parameters

1. H3-OPEN-FORM

```
1 H3METHOD (A1)        /* i /m : 'G' GET
*                      /*      : 'P' POST
1 H3URL    (A) DYNAMIC /* i /m : URL to be called
```

2. H3-CLOSE-FORM

```
                        /* none
```

How To Invoke

```
PERFORM H3-OPEN-FORM H3METHOD H3URL
PERFORM H3-CLOSE-FORM
```

48 HTML Document

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-OPEN-HTML H3-OPEN-HTML-JAVASCRIPT H3-CLOSE-HTML	H3OHTML H3CHTML	E3HTML	E3HTML

Description

Creates an HTML document with a head, title and beginning of body.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

1. H3-OPEN-HTML

```
<HTML>
<HEAD>
<TITLE>TITLE</TITLE>
</HEAD>
<BODY BACKGROUND="URL", BGCOLOR="#RPG">
```

2. H3-OPEN-HTML-JAVASCRIPT

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE='JavaScript' SRC='URL'></SCRIPT>
<SCRIPT LANGUAGE='JavaScript'>
<!--- hide script from old browsers
PROGRAM
// end hiding from old browsers -->
```

```
</SCRIPT>
<TITLE>TITLE</TITLE>
</HEAD>
<BODY BACKGROUND="URL", BGCOLOR="#RGB">
```

3. H3-CLOSE-HTML

```
</BODY>
</HTML>
```

Supported Attributes

```
BACKGROUND=URL, BGCOLOR=#RGB, SRC='URL'
```

Parameters

1. H3-OPEN-HTML

```
1 H3TITLE       (A) DYNAMIC     /* i /m: Title of the HTML document
1 H3BGCOLOR     (A) DYNAMIC     /* i /M: Background colour
1 H3BGPICTURE   (A) DYNAMIC     /* i /M: Background picture
```

2. H3-OPEN-HTML-JAVASCRIPT

```
1 H3TITLE       (A) DYNAMIC     /* i /m: Title of the HTML document
1 H3BGCOLOR     (A) DYNAMIC     /* i /M: Background colour
1 H3BGPICTURE   (A) DYNAMIC     /* i /M: Background picture
1 H3JAVASRC     (A) DYNAMIC     /* i /M: ULR to a JavaScript source
1 H3JAVA        (A/1:V) DYNAMIC /* i /M: JavaScript
1 H3JAVACOUNTER (I4)            /* i /M: Number of JavaScript source lines
1 H3ONLOAD      (A) DYNAMIC     /* i /M: onload event handler
1 H3ONUNLOAD    (A) DYNAMIC     /* i /M: onunload event handler
```

3. H3-CLOSE-FORM

```
                                /* none
```

How To Invoke

```
PERFORM H3-OPEN-HTML H3TITLE H3BGCOLOR H3BGPICTURE
PERFORM H3-OPEN-HTML-JAVASCRIPT H3TITLE H3BGCOLOR H3BGPICTURE H3JAVASRC H3JAVA
H3JAVACOUNTER H3ONLOAD H3ONUNLOAD
PERFORM H3-CLOSE-HTML
```

49 List

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-OPEN-LIST H3-LIST-ITEM H3-CLOSE-LIST	H3OLIST H3LISTI H3CLIST	E3LIST	E3LIST

Description

Creates various types of lists. Possible types are:

- unordered list,
- ordered list,
- menu-item list and
- directory list.

Cascading lists of up to 10 levels are supported. It is also possible to close more than one level at once.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

1. H3-OPEN-LIST

   `<DIR>` or
   `<MENU>` or
   `<OL>` or
   `<UL>`

2. H3-LIST-ITEM

`<LI>`

3. H3-CLOSE-LIST

`</DIR>` or
`</MENU>` or
`</OL>` or
`</UL>`

Supported Attributes

`TYPE=`*`disc`*`|`*`square`*`|`*`circle`* `TYPE=`*`1`*`|`*`a`*`|`*`A`*`|`*`i`*`|`*`I`*

Parameters

1. H3-OPEN-LIST

```
1 H3TYPE   (A1) /* i /m: Set list as:
*               /*       'O' ordered list
*               /*       'U' unordered list
*               /*       'D' directory list
*               /*       'M' menu list
1 H3BULLET (A1) /* i /m: Type of list if ordered list:
*               /*       '1' Arabic numbers (default) (1, 2, 3, ...)
*               /*       'a' Alphanumeric, lowercase (a, b, c, ...)
*               /*       'A' Alphanumeric, uppercase (A, B, C, ...)
*               /*       'i' Roman numbers, lowercase (i, ii, iii, ...)
*               /*       'I' Roman numbers, uppercase (I, II, III, ...)
*               /* i /m: Type of bullet if unordered list:
*               /*       'D' Disc
*               /*       'S' Square
*               /*       'C' Circle
```

2. H3-LIST-ITEM

```
1 H3VALUE (A) DYNAMIC /* i /m: Item text
```

3. H3-CLOSE-LIST

```
1 H3LEVEL (N2)        /* i /m: Levels to be closed
```

How To Invoke

```
PERFORM H3-OPEN-LIST H3TYPE H3BULLET
PERFORM H3-LIST-ITEM H3VALUE
PERFORM H3-CLOSE-LIST H3LEVEL
```

50 Paragraph

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-PARAGRAPH	H3PARAGR	E3PARAGR	E3PARAGR

Description

Creates a new paragraph.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

`<P ALIGN="left|right|center">` or `<P>`

Supported Attributes

`ALIGN=left|right|center`

Parameters

```
1 H3ALIGN (A1)        /* i /m : Align the paragraph to:
*                     /*        'L' Left (default)
*                     /*        'R' Right
*                     /*        'C' Center
1 H3HTML  (A) DYNAMIC /* i /mh: HTML text after the paragraph
```

How To Invoke

```
PERFORM H3-PARAGRAPH H3ALIGN H3HTML
```

51 Radio Button Group

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-RADIO-GROUP	H3RBGROU	E3RBGROU	E3RBGROU

Description

Creates a radio button group. The group can be formatted inside a table.

Appearance

`<INPUT TYPE=RADIO NAME=`*`string`*`>`

Supported Attributes

`VALUE=`*`string`*`, CHECKED`

Parameters

```
1 H3ARRAYCOUNT (I4)              /* i /m : Number of group elements
1 H3NAME       (A) dynamic       /* i /m : Name of the group variable
1 H3ARRAYVALUE (A/1:V) dynamic   /* i /M : Default value of the default group
*                                /*        variable
1 H3ARRAYLABEL (A/1:V)           /* i /mH: Label of the group element
1 H3ISCHECKED  (I4)              /* i /M : Number of default selected
*                                /*        button
1 H3LINEBREAK  (L)               /* i /M : Set line breaks between
*                                /*        buttons
1 H3ROW        (N4)              /* i /m : Set number of rows for tables
1 H3COLUMN     (N4)              /* i /m : Set number of columns for
*                                /*        tables
```

How To Invoke

```
PERFORM H3-RADIO-GROUP H3ARRAYCOUNT H3NAME
        H3ARRAYVALUE(*) H3ARRAYLABEL(*)
        H3ISCHECKED H3LINEBREAK H3ROW H3COLUMN
```

52 Horizontal Rule

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-RULE	H3RULE	E3RULE	E3RULE

Description

Creates a horizontal rule with a width specified in percentage points.

Appearance

<HR> or
<HR WIDTH="*p%*">

Supported Attributes

WIDTH=*p%*

Parameters

```
1 H3WIDTH (N4) /* i /m : Width in percent
```

How To Invoke

```
PERFORM H3-RULE H3WIDTH
```

53 Scrolling List

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-SCROLLING-LIST	H3SCLIST	E3SCLIST	E3SCLIST

Description

Creates a scrolling list. It can be displayed as a combo box or as a list box. Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

```
<SELECT NAME=string>
<OPTION> </OPTION>
</SELECT>
```

Supported Attributes

```
SIZE=n, MULTIPLE

VALUE=string, SELECTED
```

Parameters

```
1 H3SIZE           (N4)             /* i /m : Number of lines:
*                                   /*        =1 combo box
*                                   /*        >1 list box
1 H3ARRAYCOUNT     (I4)             /* i /m : Number of list elements
1 H3NAME           (A)dynamic       /* i /m : Name of the group variable
1 H3ARRAYVALUE     (A/1:V) dynamic  /* i /M : Default value of the list values
1 H3ARRAYLABEL     (A/1:V) dynamic  /* i /MH: Label of the list elements
1 H3ARRAYSELECTED  (L/1:V)          /* i /M : Elements selected by
*                                   /*        default
1 H3MULTIPLE       (L)              /* i /M : Multiple selection allowed
```

How To Invoke

```
H3-SCROLLING-LIST H3SIZE H3ARRAYCOUNT H3NAME
      H3ARRAYVALUE(*) H3ARRAYLABEL(*)
      H3ARRAYSELECTED(*) H3MULTIPLE
```

54 Table

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-TABLE H3-TABLE-COLOR	H3TABLE H3TABLEC	E3TABLE	E3TABLE

Description

Creates a simple table for a given array. With `H3-TABLE-COLOR`, for headline and table, different background colors can be set. The rows will be colored alternately.

Appearance

```
<TABLE>
<TR>
<TH>... </TH>
</TR>
...
<TR>
<TD>... </TD>
</TR>
...
</TABLE>
```

Supported Attributes

`ALIGN=`*`left`*`|`*`right`*`|`*`center`* `, BORDER=`*`n`*`, NOWRAP`

Parameters

1. H3-TABLE

```
1 H3ROW           (N4)                 /* i /m : Number of rows
1 H3COLUMN        (N4)                 /* i /m : Number of columns
1 H3ARRAY2VALUE   (A/1:V,1:V)dynamic   /* i /mh: Table elements
1 H3ARRAY2ALIGN   (A1/1:V,1:V)         /* i /M : Alignment of the table cells
*                                      /*        'L' Left (default)
*                                      /*        'R' Right
*                                      /*        'C' Center
1 H3ARRAY2NOWRAP  (L/1:V,1:V)          /* i /m : No automatic wrapping
1 H3HEADLINE      (L)                  /* i /M : 1st line as headline
1 H3ALIGN         (A1)                 /* i /M : Alignment of the table
*                                      /*        'L' Left (default)
*                                      /*        'R' Right
*                                      /*        'C' Center
1 H3SUPPRESSEMPTY (L)                  /* i /m : Set to TRUE if cell is
*                                      /*        to be displayed
*                                      /*        despite being empty
1 H3ISHTML        (L)                  /* i /m : Transform value to
*                                      /*        HTML
1 H3BORDER        (N4)                 /* i /M : Set border size
```

2. H3-TABLE-COLOR

```
1 H3ROW         (N4)                /* i /m : Number of rows
1 H3COLUMN      (N4)                /* i /m : Number of columns
1 H3TITLECOLOR  (A032)              /* i /M : Color of headline
1 H3LINECOLOR   (A032)              /* i /M : Color of lines
1 H3ARRAY2VALUE (A/1:V,1:V) dynamic /* i /mh: Table elements
1 H3ARRAY2ALIGN (A001/1:V,1:V)      /* i /m : Alignment of the table cells
*                                   /*        'L' Left (default)
*                                   /*        'R' Right
*                                   /*        'C' Center
1 H3ARRAY2NOWRAP (L/1:V,1:V)        /* i /m : No automatic wrapping
1 H3HEADLINE     (L)                /* i /m : 1st line as headline
1 H3ALIGN        (A1)               /* i /M : Alignment of the table
*                                   /*        'L' Left
*                                   /*        'R' Right
*                                   /*        'C' Center (default)
1 H3SUPPRESSEMPTY (L)               /* i /m : Set to TRUE if cell is
*                                   /*        to be displayed
*                                   /*        despite being empty
1 H3ISHTML        (L)               /* i /m : Transform value to
*                                   /*        HTML
1 H3BORDER        (N4)              /* i /m : Set border size
```

How To Invoke

```
PERFORM H3-TABLE H3ROW H3COLUMN H3ARRAY2VALUE(*,*)
      H3ARRAY2ALIGN(*,*) H3ARRAY2NOWRAP(*,*)
      H3HEADLINE H3ALIGN H3SUPPRESSEMPTY
      H3ISHTML H3BORDER

PERFORM H3-TABLE-COLOR H3ROW H3COLUMN H3TITLECOLOR H3LINECOLOR
       H3ARRAY2VALUE(*,*) H3ARRAY2ALIGN(*,*) H3ARRAY2NOWRAP(*,*)
       H3HEADLINE H3ALIGN H3SUPPRESSEMPTY
       H3ISHTML H3BORDER
```

55 Universal Tag

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-TAG	H3TAG	E3TAG	E3TAG

Description

Creates a universal tag (tag template) inside an HTML page. This tag template creates the framework into which code can be written.

Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

<tag> </tag>

Parameters

```
1 H3PRE  (A) DYNAMIC /* i /m : Open tag value
1 H3HTML (A) DYNAMIC /* i /m : HTML inside the tag
1 H3POST (A) DYNAMIC /* i /m : Close tag value
```

How To Invoke

```
PERFORM H3-TAG H3PRE H3HTML H3POST
```

56 Text Area

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-TEXT-AREA	H3TXAREA	E3TXAREA	E3TXAREA

Description

Creates a text area. Changes from previous versions

All (A250) BY VALUE variables at the interfaces have been changed to (A) DYNAMIC BY VALUE.

There is no need to recatalog the programs, because the old interface (A250) is compatible with the new one.

Appearance

```
<TEXTAREA NAME="string"> </TEXTAREA>
```

Supported Attributes

```
ROWS=n, COLS=n
```

Parameters

```
1 H3ARRAYCOUNT (I4)            /* i /m : Number of text
1 H3NAME       (A) dynamic     /* i /m : Name of the text variable
1 H3ARRAYTEXT  (A/1:V) dynamic /* i /M : Default value of the text variable
1 H3ROW        (N4)            /* i /M : Set number of rows
1 H3COLUMN     (N4)            /* i /M : Set number of columns
```

How To Invoke

```
PERFORM H3-TEXT-AREA H3ARRAYCOUNT H3NAME H3ARRAYTEXT(*)
   H3ROW H3COLUMN
```

57 Text to URL - Decoded

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-TEXT-TO-URL H3-ASCII-URL-TABLE	H3TX2URL	E3TX2URL	E3TX2URL

Description

Converts a string to URL-decoded syntax. Useful if special characters are included. Use `W3-TEXT-TO-URL` instead of H3-TEXT-TO-URL. Use `W3-ASCII-URL-TABLE` instead of H3-ASCII-URL-TABLE.

The subprogram W3-ASCII-URL-TABLE will be called from H3-TEXT-TO-URL and contains a list of all conversations that will be made. This program can be changed and extended for the user's needs.

Parameters

```
1 H3COUNT  (I4)   /*  o/m : Length of the converted string
1 H3STRING (A250) /* io/m : URL-decoded text after conversion
```

How To Invoke

```
PERFORM H3-TEXT-TO-URL H3COUNT H3STRING
```

58 Time/Date String

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-TIME_DATE H3-TIME-DATE	H3TIMDAT H3TIMDA	E3TIMDAT	E3TIMDAT

Description

Creates a 'generated: ...' string using the LOG time or an HTTP-compatible time/date string with offset, using the current time/date (because GMT or offset to GMT is not known by Natural).

Appearance

`generated: time/date`

Parameters

1. H3-TIME_DATE

```
/* none
```

2. H3-TIME-DATE

```
1 H3ADDMINUTE (I4)  /* i /m : Adds minutes to time
1 H3ADDDAY    (I4)  /* i /m : Adds days to date
1 H3DATETIME  (A29) /*  o/m : Generated string
```

How To Invoke

```
PERFORM H3-TIME_DATE
PERFORM H3-TIME-DATE H3ADDMINUTE H3ADDDAY H3DATETIME
```

59 List all Natural Libraries

Subprogram Name	Executable Example	Viewable Example
NAT-LIB	NAT-LIB NAT-LIB?FNAT=N	NAT-LIB

Description

Generates an HTML page and displays all available Natural libraries. If no FNAT parameter is given, the default user libraries will be displayed.

Parameters

FNAT=	N = system libraries U = user libraries (default)
EXPIRE=	Adds days to current date and sets it as expiry date.
START=	Wildcard selection for the displayed object set.

How To Invoke

```
NAT-LIB NAT-LIB?FNAT=N
```

60 Run Online Natural Web Interface Subprograms

Natural Program
WEB-ONL

Description

For reasons of debugging or testing, it is useful to run Natural Web Interface subprograms online. The output of the generated page will be saved as a Natural text object. Lines longer than 92 characters will be split. It is possible to set environment variables. If the variables should be set as server variables, add an ampersand in front of the name.

How To Invoke

Run Program `WEB-ONL` from the Natural *next* prompt.

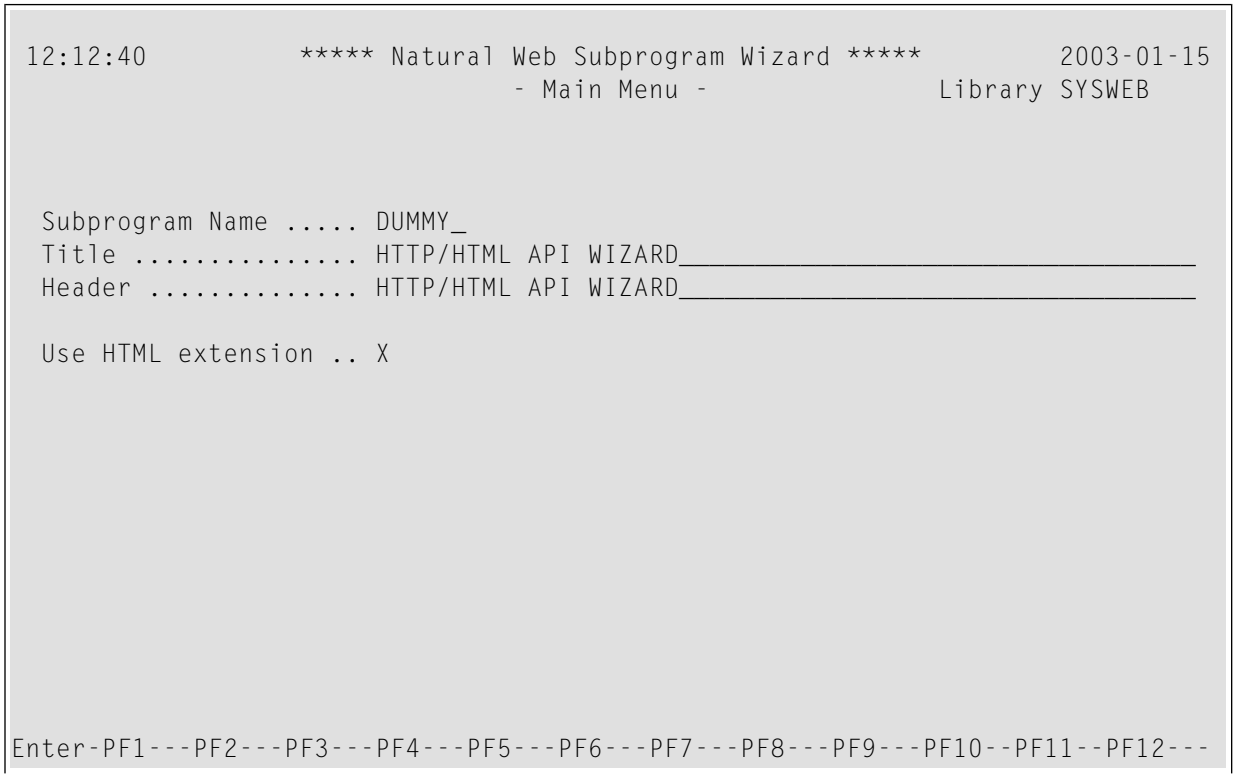
61 Generate Natural Subprogram to use with Natural Web Interface

Natural Program	Executable Generation Result	Viewable Generation Result
WEB-WIZ	Basic Subroutines HTML Extension	HTTPApi HTMLApi

Description

Generates a default program. This function will be deleted in one of the next versions. .

Input Map



```
                    Exit
```

How To Invoke

Run Program `WEB-WIZ` from the Natural *next* prompt.

62 List Directory of a Natural Library

Subprogram Name	Executable Example	Viewable Example
NAT-DIR	NAT-DIR?lib=sysweb NAT-DIR?lib=sysweb&start=E3*	NAT-DIR

Description

Generates an HTML page with the directory information of a Natural library. If no library parameter has been defined, the current library will be displayed.

Parameters

LIB=	Natural LIBRARY
EXPIRE=	Adds days to current date and sets it as expiry date.
START=	Wildcard selection for the object set displayed.

How To Invoke

```
nat-dir?lib=sysweb
```

63 List All Parameters Passed From a HTTP Server To a Called Natural Subprogram

Subprogram Name	Executable Example	Viewable Example
NAT-ENV	NAT-ENV	NAT-ENV

Description

Generates an HTML page with all parameters passed from the HTTP server to a called Natural subprogram.

How To Invoke

```
nat-env
```

64 Return an HTML Page Saved as Natural Source Object

Subprogram Name	Executable Example	Viewable Example	Text Object
NAT-HTML	NAT-HTML?lib=sysweb&source=t3-html	NAT-HTML	T3-HTML

Description

Displays an HTML page saved as a Natural source object.

Parameters

LIB=	Natural LIBRARY
SOURCE=	SOURCE NAME

How To Invoke

```
nat-html?lib=sysweb&source=HTML
```

65 List the Current Natural Web Interface Settings

Subprogram Name	Executable Example	Viewable Example
NAT-INFO	NAT-INFO	NAT-INFO

Description

Generates an HTML page with information about your HTTP Browser, HTTP Server, communication software (RPC/DCOM) and Natural environment.

How To Invoke

```
nat-info
```

66 List Source of Natural Object

Subprogram Name	Executable Example	Viewable Example
NAT-LIST	NAT-LIST?lib=sysweb&source=h3image	NAT-LIST

Description

Generates an HTML page with the listing of a Natural source object.

Parameters

LIB=	Natural LIBRARY
SOURCE=	SOURCE NAME
EXPIRE=	Adds days to current date and sets it as expiry date.
LINE-NUMBERS=	The only value possible is OFF

How To Invoke

```
nat-list?lib=sysweb&source=H3IMAGE
```

67 Online Documentation

Subprogram Name	Executable Example
NAT-DOCU	NAT-DOCU

Description

Displays the online documentation saved as Natural source objects.

Parameters

LIB=	Natural LIBRARY
SOURCE=	SOURCE NAME
EXPIRE=	Adds days to current date and sets it as expiry date.

How To Invoke

```
nat-docu
```

68 XML Toolkit

Preface

XML, the eXtensible Markup Language, has become one of the most relevant standards and a driving force for Web applications. As of Natural Version 4 for mainframes , XML documents can be created, processed and made available. Because complete highly nested XML documents can be very large, Natural's data-definition and data-manipulation capabilities have been extended with large (up to one gigabyte) and dynamic variables.

This Document

The XML Toolkit provides Natural with XML document processing capabilities. A Natural data definition can be generated from an XML Document Type Definition (DTD), and vice versa. The content of a Natural variable can be serialized into an XML document. And an XML document can be parsed into a Natural variable.

This document describes an example application that demonstrates the use of XML within a Natural-for-Mainframes environment without external program parts.

The following topics are covered:

- **Examples**
- **Parser Error Messages**

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

69 Introduction

The following topics are covered:

XML Toolkit Features

- Natural-based XML parser using dynamic variables.
- Functions for
  - conversion of Natural data structures into DTD definitions;
  - generation of COMPRESS statements to save a Natural data structure as an XML document;
  - generation of callback for the Natural-based parser.

XML Toolkit Description

Objective

The objective of the Natural XML Toolkit is to provide additional XML functionality with Natural and improve the integration of Natural applications with XML.

General Architecture

The Natural XML Toolkit consists of a collection of Natural programs. The Toolkit programs may be integrated into customer applications to provide access to XML data or to deliver data from Natural in XML format.

The Natural XML Toolkit calls the functions listed below:

XML Toolkit Functions

1. Mapping of Natural Data Definition to DTD and vice versa.
2. **XML Token => NAT**
   Data After the Natural data structure has been created, the XML document has to be parsed and saved into the data structure. A Natural implementation is generated that is capable of saving the given data into the Natural data structure.
3. **NAT Data => XML Document ("Serialize")**
   Serialization is the process of taking the data stored in the Natural data structures and creating an XML document according to the description provided.

A Natural dialog implements the user interface to the XML Toolkit functions. The DTD will be accessed as a work file and the generated Natural objects will be saved directly to the Natural system file.

Map Natural Data Definitions to DTD

This mapping is the first step to bind Natural data structures to XML tags and is required to implement a representation of Natural data as XML tags. The example below shows the mapping as well as some obvious differences between Natural and a DTD.

Natural PDA

```
                      Press ESC to enter command mode
 Mem: EMPL     Lib: SYSEXXT  Type: PARAMETER  Bytes:  1072  Line:    0 of:  26
C T   Comment
  *   *** Top of Data Area ***
    1 EMPLOYEE
    2 ATTRIBUTES_OF_EMPLOYEE
    3 PERSONNEL-ID                    A          8
  *
    2 FULL-NAME
    3 FIRST-NAME                      A         20
    3 NAME                            A         20
  *
    2 FULL-ADDRESS
    3 C@ADDRESS-LINE                  I          4
    3 ADDRESS-LINE                    A         20 (1:6)
    3 CITY                            A         20
    3 ZIP                             A         20
    3 COUNTRY                         A          3
  *
    2 TELEPHONE
    3 AREA-CODE                       A          6
    3 PHONE                           A         15
```

Generated DTD

```
<!ELEMENT EMPLOYEE (PERSONNEL-ID, FULL-NAME, FULL-ADDRESS, TELEPHONE, INCOME* )>

<!ELEMENT PERSONNEL-ID (#PCDATA ) >

<!ELEMENT FULL-NAME (FIRST-NAME, NAME )>
         <!ELEMENT FIRST-NAME  (#PCDATA )>
         <!ELEMENT NAME  (#PCDATA )>

<!ELEMENT FULL-ADDRESS (ADDRESS-LINE*, CITY, ZIP, COUNTRY )>
         <!ELEMENT ADDRESS-LINE  (#PCDATA )>
         <!ELEMENT CITY  (#PCDATA )>
         <!ELEMENT ZIP  (#PCDATA )>
         <!ELEMENT COUNTRY  (#PCDATA )>
...
```

The generated DTD will be used later on during serialization to a XML document (see below).

Serialize Data to XML

During execution of a Natural program, the content of the data defined in the DEFINE DATA statement will be filled with "real" content. This content will be written to a dynamic variable in XML format during serialization and will use the formerly generated DTD as input.

The XML Toolkit generates the program to serialize the data.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<EMPLOYEE PERSONNEL-ID="30016509">
<FULL-NAME>
  <FIRST-NAME>ELSPETH</FIRST-NAME>
  <NAME>TROWBRIDGE</NAME>
</FULL-NAME>
<FULL-ADDRESS>
  <ADDRESS-LINE>91 BACK LANE</ADDRESS-LINE>
  <ADDRESS-LINE>BILSTON</ADDRESS-LINE>
  <ADDRESS-LINE>STAFFORDSHIRE</ADDRESS-LINE>
  <CITY>BILSTON</CITY>
  <ZIP>ST2 3KA</ZIP>
  <COUNTRY>UK</COUNTRY>
</FULL-ADDRESS>
<TELEPHONE>
  <PHONE>863322</PHONE>
  <AREA-CODE>0602</AREA-CODE>
</TELEPHONE>
...
```

Map DTD to Natural Data Definitions

The mapping of a DTD to Natural data structures again shows differences. The DTD does not specify how many person records will be included in the XML document, therefore the Toolkit assumes that a maximum number of "v" persons will be included. The application programmer might know the exact number and the data structure could be adapted accordingly. A similar limitation exists with the length of the data. The DTD does not include information about the length of the data in a person's record. Therefore the Toolkit creates fields in the data structure with a length of 253, the current maximum.

```
* DTD E:\SAG\nat\4.2\fnat\SYSEXXT\RES\empl.dtd
COMPRESS &1& '<EMPLOYEE'
  ' PERSONNEL-ID="'EMPLOYEE.PERSONNEL-ID "'
  '>' INTO &1& LEAVING NO
/* now the children
COMPRESS &1& '<FULL-NAME'
  '>' INTO &1& LEAVING NO
/* now the children
COMPRESS &1& '<FIRST-NAME'
  '>'
  EMPLOYEE.FIRST-NAME
  '</FIRST-NAME>' INTO &1& LEAVING NO
COMPRESS &1& '<NAME'
  '>'
  EMPLOYEE.NAME
  '</NAME>' INTO &1& LEAVING NO
/*
COMPRESS &1& '</FULL-NAME>' INTO &1& LEAVING NO
COMPRESS &1& '<FULL-ADDRESS'
  '>' INTO &1& LEAVING NO
/* now the children
FOR &2& = 1 TO EMPLOYEE.C@ADDRESS-LINE
  COMPRESS &1& '<ADDRESS-LINE'
    '>'
    EMPLOYEE.ADDRESS-LINE(&2&)
    '</ADDRESS-LINE>' INTO &1& LEAVING NO
END-FOR
...
```

Parse XML File and Assign to Natural Data

```
* DTD E:\SAG\nat\4.2\fnat\SYSEXXT\RES\empl.dtd
DECIDE ON FIRST &1&
  VALUE 'EMPLOYEE'
    RESET INITIAL EMPLOYEE
  VALUE 'EMPLOYEE/@PERSONNEL-ID'
    /* #REQUIRED
    EMPLOYEE.PERSONNEL-ID := &3&
  VALUE 'EMPLOYEE/FULL-NAME'
    IGNORE
  VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME'
    IGNORE
  VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME/$'
    EMPLOYEE.FIRST-NAME := &3&
  VALUE 'EMPLOYEE/FULL-NAME/NAME'
    IGNORE
  VALUE 'EMPLOYEE/FULL-NAME/NAME/$'
    EMPLOYEE.NAME := &3&
```

```
...
```

Outlook

The XML Toolkit is another step forward to full XML support with Natural. The XML Toolkit might be extended after the first release. However, the main objective is to implement XML functionality in one of the forthcoming releases as part of Natural's powerful language.

Considerations and Limitations

When using the XML Toolkit, the following limitations should be considered.

Very Large Data Structures

Data structures which will result in more then approximately 700 data fields and groups will end up with the message:

```
Input Structure too big
```

Solution

Split up the data structure into smaller sections.

70 Using the XML Toolkit

The following topics are covered:

Prerequisites

Storage Requirements

Depending on the size, complexity or the recursion depth of the processed XML DTD, the XML Toolkit might require up to several hundred kilobytes of space in the DATSIZE buffer.

Parsing an XML document using the generated callback routine requires that the entire document is contained in a dynamic variable.

Schema Support

Schemas are not supported by the XML Toolkit on mainframe computers; only DTDs can be processed.

Codepage Support

For the use of the XML Toolkit, the Natural session must be driven with code page support enabled. See Natural profile parameter `CP`.

Work File Processing

By default, the XML Toolkit uses Natural Work File 12 as input or output file for DTDs and Work File 13 as error logfile.

The sample parser program, delivered with the XML Toolkit, reads the XML document to be parsed from Work File 12 and writes the output of the parse process to Work File 13.

Work File Support

Running the XML Toolkit under TP monitors which do not support Natural work files is possible using PC work files, if you are working with a PC on which Entire Connection is in use.

XML Documents which shall be parsed using the generated callback routine or which shall be generated using the generated serialization code can be accessed via work files as well. Data must be accessed with a variable of data type ALPHA DYNAMIC. The work file should be of data type UNFORMATTED.

PC files can be accessed using the subprogram XML2PCWR provided in the SYSEXXT library. XML2PCWR writes or reads an ALPHA DYNAMIC variable to or from a PC work file.

Calling Conventions for XML2PCWR:

```
CALLNAT 'XML2PCWR' XML-PAGE FILENUMBER OPERATION RETCODE
```

Parameter Definition:

```
DEFINE DATA LOCAL
  1 XML-PAGE            (A)     DYNAMIC         /* XML page
  1 FILENUMBER          (I2)                    /* work file number: 1 - 31
  1 OPERATION           (A2)                    /* W/R: write/read
  1 RETCODE             (I4)                    /* must be 0!
END-DEFINE
```

Example:

The following example writes the generated document to Standard Work File 10 and to PC Work File 15:

```
* ------------------------------------------------------------------------
* CLASS NATURAL XML TOOLKIT - UTILITIES
*
* SDEMO_P1
*
* DESCRIPTION
* Serialize a given Data structure.
*
*
* AUTHOR SAG 01.2005
*
* VERSION 4.12.
*
* (c) Copyright Software AG 2001-2005. All rights reserved.
*
* ------------------------------------------------------------------------
*
DEFINE DATA
LOCAL USING EMPL /* add generated data structure
LOCAL
1 XML (A) DYNAMIC
*
1 OUT (A72)
1 II (I4)
*
1 #CX (I4)
1 #CY (I4)
1 #CZ (I4)
1 FILENUMBER (I2)
```

```
1 OPERATION (A2)
1 RETCODE (I4)
END-DEFINE
/*[ initialize
EMPLOYEE.PERSONNEL-ID := 4711
*
EMPLOYEE.FIRST-NAME := "ADKINSON"
EMPLOYEE.NAME := "MARTHA"
*
EMPLOYEE.CÂ§ADDRESS-LINE := 2
EMPLOYEE.ADDRESS-LINE(1) := "8603 GARLAND COURT"
EMPLOYEE.ADDRESS-LINE(2) := "FRAMINGHAM"
EMPLOYEE.ADDRESS-LINE(2) := "MA"
EMPLOYEE.CITY := "FRAMINGHAM"
EMPLOYEE.ZIP := "17010"
EMPLOYEE.COUNTRY := "USA"
*
EMPLOYEE.AREA-CODE := "617"
EMPLOYEE.PHONE := "210-4703"
*
EMPLOYEE.JOB-TITLE := "MANAGER"
EMPLOYEE.CÂ§INCOME := 2
EMPLOYEE.SALARY(1) := 47000
EMPLOYEE.CÂ§BONUS(1) := 2
EMPLOYEE.BONUS(1,1) := 10500
EMPLOYEE.BONUS(1,2) := 7875
*
EMPLOYEE.SALARY(2) := 47000
EMPLOYEE.CÂ§BONUS(2) := 1
EMPLOYEE.BONUS(2,1) := 35700
*
INCLUDE EMPL-C "XML" "#CX" "#CY" "#CZ"
/* add generated Serialize
/*]
*
ASSIGN FILENUMBER = 15 /* PC FILE
ASSIGN OPERATION = 'W'
CALLNAT 'XML2PCWR' XML FILENUMBER OPERATION RETCODE
PRINT XML
DEFINE WORK FILE 10 TYPE 'UNFORMATTED' /* STD WORK FILE
WRITE WORK FILE 10 VARIABLE XML
CLOSE WORK FILE 10
*
END
```

Print Files

The XML Toolkit writes to Report 2.

Invoking the Application

The XML toolkit is included in the library SYSEXXT.

To use the XML Toolkit

- In the Natural command line, enter LOGON SYSEXXT.

  Enter Menu.

  The Main Menu is displayed.

```
10:22:54                *** NATURAL XML Toolkit ***               2007-01-19
                              - Main Menu -                    Library SYSEXXT




                      Code  Function


                        L   Generate from Natural Data Structure
                        D   Generate from Document Type Definition
                        O   Set up Specific Generation Options


                      Function Code .. _




 Press PF4 or PF5 to start generation.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  LDA   DTD                           Optio       Canc
```

  The following functions are available:

- **Generate from Natural Data Structure** Uses the Natural Data Area as a data source.
- **Generate from Document Type Definition** Uses the Document Type Definition (.dtd) as a data source.
- **Set up Specific Generation Options**

For details, refer to the corresponding sections.

PF-Key Assignments

The following function keys are used for navigation and processing.

PF1	Help	Context-related help. For more information, refer to the online documentation.
PF3	Exit	On an options map: closes the function and saves the changes. On a generation map: closes the function after a generation is done. On the Main Menu map: closes the application.
PF7	Prev	Previous step (previous map).
PF8	Next	Next step (next map).
PF9	Finis(h)	Closes the function after a generation is done.
PF12	Cancel	Closes the function without saving the changes. Closes the function if no generation has already be done.

71 Setting up Specific Generation Options

The generation options are arranged on on two screens and are grouped into data fields and path definition.

The following topics are covered below:

Invoking the Generation Options Setup Screens

To invoke the first screen of the generation options function

- On the Main Menu screen, press PF10 **Optio**(n).

  The first screen of the Options setup function is displayed. The map fields are described below.

First Screen

Special characters that are not valid in XML have to be converted into valid names. The following map enables you to change the default conversion settings, if required.

```
09:51:35                *** NATURAL XML Toolkit ***                2007-01-17
User DEFAULT                     - Options -                     Library SYSEXXT


 Additional fields
 Counter separator character ............. @

 XML name replacements
 Namespace separator character ':' with .. $
 Dot sign '.' with ....................... /

 Natural variable name replacements
 Plus sign '+' with ...................... plus_______
 Hash / Number sign '#' with ............. hash_______
 Slash sign '/' with ..................... slash______
 At sign '@' with ........................ at_________
 Paragraph sign 'Â§' with ................. paragraph__
 Ampersand sign '&' with ................. ampersand__
 Dollar sign '$' with .................... dollar_____


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

Field Descriptions

Counter Separator Character

Belongs to Group:	Additional Fields
Format/Length:	A1
Default Value:	@

Namespace Separator Character ":" WITH:

Belongs to Group:	XML Name Replacements
Format/Length:	A1
Default Value:	$

Dot Sign '.' WITH:

Belongs to Group:	XML Name Replacements
Format/Length:	A1
Default Value:	/

Plus Sign '+' WITH:

Belongs to Group:	XML Variable Name Replacements
Format/Length:	A11
Default Value:	plus

Hash / Number Sign '#' WITH:

Belongs to Group:	Natural Variable Name Replacements
Format/Length:	A11
Default Value:	hash

Slash Sign '/' WITH:

Belongs to Group:	Natural Variable Name Replacements
Format/Length:	A11
Default Value:	slash

At Sign '@' WITH:

Belongs to Group:	Natural Variable Name Replacements
Format/Length:	A11
Default Value:	at

Paragraph Sign '§' WITH:

Belongs to Group:	Natural Variable Name Replacements
Format/Length:	A11
Default Value:	para

Ampersand Sign '&' WITH:

Belongs to Group:	Natural Variable Name Replacements
Format/Length:	A11
Default Value:	amp

Dollar Sign '$' WITH:

Belongs to Group:	Natural Variable Name Replacements
Format/Length:	A11
Default Value:	dollar

Second Screen

The second map of the Options function serves to define the location of the target or source DTD file used for the conversion.

To invoke the second screen of the Generation Options function

1 On the Main Menu screen, press PF10 **Optio**(n).

2 On the first screen, press PF8 **Next**.

The second screen of the Options function is displayed. The map fields are described below.

```
11:04:10                *** NATURAL XML Toolkit ***                2007-01-17
User DEFAULT                     - Options -                    Library SYSEXXT


 External file:
 /nat_64/proj/natc/42/samples/sysexxt/empl.dtd_____________________________________

 Natural library:
 SYSEXXT_









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

Field Descriptions

External file

Format/Length:	A253
Default Value:	(Resource directory of current library)

Natural library

Format/Length:	A8
Default Value:	(current library)

Saving Your Options Permanently

All settings made in the Options menu are written to the text member XML-INI.

Whenever a new Natural patch level or service pack is installed, all settings made in the Options menu are overwritten.

In order to keep your settings permanently, you are recommended to save the text member XML-INI to the FUSER before you install a patch level or service pack.

72 Using a Natural Data Source

This function enables you to generate an XML document from a data definition held in a Natural local, global or parameter data area.

The following topics are covered:

See also:

- **Using a Document Type Definition as Data Source**
- **Setting up Specific Generation Options**
- **PF-Key Assignments**

Select Natural Data Area

This screen serves to select LDA, GDA or PDA as input Data Area.

Note: The entries shown in the dialogs below are default or example values.

```
08:07:48              ***** NATURAL XML Toolkit *****              2007-01-18
                - Generate from Natural Data Structure -       Library SYSEXXT




 Select LDA, GDA or PDA as input Data Area.

 Press 'Next' to read the Data Area.

 Select a Level 1 group that should be used for further generations.


 Select Data Area for generation
 Library:  Type: Name:
 SYSEXXT_  L     ________









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

Field Descriptions

Library

Belongs to Group:	Select Input Data Area
Format/Length:	A8
Default Value:	(All libraries)

Type

Belongs to Group:	Select Input Data Area
Format/Length:	A1
Possible Values:	**L -** Local Data Area **A -** Parameter Data Area **G -** Global Data Area

Name

Belongs to Group:	Select Input Data Area
Format/Length:	A8
Default Value:	(All objects of the selected library and type)

Press PF8 to continue.

Select Root Group

This screen is used to select the data type.

```
10:18:34                *** NATURAL XML Toolkit ***              2007-01-18
                           - Select Root Group -              Library SYSEXXT

 Root groups
 X EMPLOYEE
 _
 _
 _
 _
 _
```

```
_
_
_
_
_
_
_
_
_
```

Field Descriptions

Format/Length:	A1
Default Value:	(All groups on Level 1)

Mark the desired element, e.g. EMPLOYEE with an **X** and press ENTER .

Press PF8 to continue.

Generate File with DTD Definition

This screen serves to generate a file with the DTD definition of a given group.

```
11:14:49              ***** NATURAL XML Toolkit *****               2007-01-18
                - Generate from Natural Data Structure -       Library SYSEXXT



  Generate file with the dtd definition of a given group.

  Specify a File Name and Press 'Next' to start the generation.

  Press 'Next' to ignore this generation.



  Select output for DTD file:
  $NATDIR/$NATVERS/42/samples/sysexxt/empl.dtd_____________________________________




 Serialize Copycode generation done.
```

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

Format/Length:	A253

Press PF8 to continue.

Generate a parser for an XML document

This screen is used to generate copycode as implementation for the serialization of the given group into an XML document.

```
10:56:06              ***** NATURAL XML Toolkit *****              2007-01-18
                 - Generate from Natural Data Structue -       Library SYSEXXT



 Generate Copycode as implemention for the serialization of the given
 group into a XML document.

 Specify a Name and Press 'Next' to start the generation.

 Press 'Next' to ignore this generation.

 Select output for Serialize Copycode
 Library:  Type: Name:
 SYSEXXT_ C     ________




Parse Copycode generation done.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                   Prev  Next                   Canc
```

Generates the parser CALLBACK copycode. See also **Parser CALLBACK Copycode** (in the Examples document).

Field Descriptions

Library

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output Copycode
Format/Length:	A21
Default Value:	Copycode

Name

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All objects of the selected library and type)

Press PF8 to continue.

Show Generation Report

After the generation is complete, the generation report is displayed.

```
11:48:29              ***** NATURAL XML Toolkit *****              2007-01-18
                - Generate from Natural Data Structure -       Library SYSEXXT

  Generation Results
  Generate for Data Area
    Library ..: SYSEXXT
    Object ...: EMPL
    Read Data Area done.
  Parser (Callback) Copycode
    Library ..: SYSEXXT
    Source ...: A1
    Parse Copycode generation done.
  Serialize (Compress XML) Copycode
    Library ..: SYSEXXT
    Source ...: A2
```

```
   Serialize Copycode generation done.
 File with dtd Definition
   File .....: $NATDIR/$NATVERS/42/samples/sysexxt/esi.dtd
   Generate DTD file.

Generation done.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                   Prev        Finis             Canc
```

Field Descriptions

Summary

Format/Length:	A253/1:v

Press PF9 **Finis**(h) to end the generation process.

73 Using an external Data Source

This function enables you to parse an XML document into a Natural variable defined in a local, global or parameter data area.

The following topics are covered:

See also:

- **Using a Natural Data Area as Data Source**
- **Setting up Specific Generation Options**
- **PF-Key Assignments**

Generate from Document Type Definition

This screen is used to select a Document Type Definition/Tamino Schema as input Document Type.

Note: The field entries shown in the screens below are default or example values.

```
12:30:09              ***** NATURAL XML Toolkit *****              2007-01-19
               - Generate from Document Type Definition -       Library SYSEXXT



 Select Document Type Definition/Tamino Schema as input Document Type.

 Press 'Next' to read the Document Type.

 Select a root element or Tamino Document Type that should be used
 for further generations.


 Select DTD/Tamino Schema for generation:
 /nat_64/proj/natc/42/samples/sysexxt/empl.dtd_____________________________




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

Field Descriptions

Input File

Format/Length:	A253

Use Tamino 2.1.x Schema instead of DTD.

Press PF8 **Next** to continue.

Select Root Element

This screen is used to select an element that should be the root of your XML document.

```
12:42:17                *** NATURAL XML Toolkit ***              2007-01-19
                          - Select Root Element -            Library SYSEXXT

 Element
 X EMPLOYEE
 _ FULL-NAME
 _ FIRST-NAME
 _ NAME
 _ FULL-ADDRESS
 _ ADDRESS-LINE
 _ CITY
 _ ZIP
 _ COUNTRY
 _ TELEPHONE
 _ AREA-CODE
 _ PHONE
 _ JOB-TITLE
 _ INCOME
 _ SALARY
```

Field Descriptions

Root Element (for DTDs)

Default Value:	(All Elements)

Mark the desired element, e.g. EMPLOYEE, with an **X** and press ENTER.

Generate Natural Data Area

This is used to generate a Natural Data Area with definition of a group that represents the XML document.

```
13:25:40              ***** NATURAL XML Toolkit *****              2007-01-19
              - Generate from Document Type Definition -       Library SYSEXXT



 Generate Data Area with definition of a group that represents
 the XML document.

 Specify a Name and Press 'Next' to start the generation.

 Press 'Next' to ignore this generation.

 Select output LDA/GDA/PDA
 Library:  Type: Name:
 SYSEXXT_ L     A3______




Generate Data Area.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                   Prev  Next                   Canc
```

Field Descriptions

Library

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output Copycode
Format/Length:	A21
Default Value:	**L** - Local Data Area

Name

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All objects of the selected library and type)

Next to continue.

Serialize into XML Document

This screen is used to generate copycode as implementation for the serialization of the given group into an XML document.

```
13:10:40              ***** NATURAL XML Toolkit *****              2007-01-19
                - Generate from Natural Data Structure -       Library SYSEXXT



 Generate Copycode as implemention for the serialization of the given
 group into a XML document.

 Specify a Name and Press 'Next' to start the generation.

 Press 'Next' to ignore this generation.

 Select output for Serialize Copycode
 Library:  Type: Name:
```

```
 SYSEXXT_ C     A2_______




Parse Copycode generation done.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                   Prev  Next                   Canc
```

See also **Serialize Copycode** (in the Examples document).

Field Descriptions

Library

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output Copycode
Format/Length:	A21
Default Value:	Copycode

Name

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All objects of the selected library and type)

Press PF8 **Next** to continue.

Generate Copycode

This screen is used to generate copycode as implementation for the XML Parser Callback for the given group.

```
13:02:32              ***** NATURAL XML Toolkit *****              2007-01-19
               - Generate from Document Type Definition -       Library SYSEXXT



 Generate Copycode as implemention for the XML Parser Callback for the
 given group.

 Specify a Name and Press 'Next' to start the generation.

 Press 'Next' to ignore this generation.

 Select output for Parse Copycode:
 Library:  Type: Name:
 SYSEXXT   C




Read DTD/Tamino Schema done.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                   Prev  Next                    Canc
```

Generates the parser CALLBACK copycode. See also **Parser CALLBACK Copycode** (in the Examples document).

Field Descriptions

Library

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output Copycode
Format/Length:	A1
Default Value:	Copycode

Name

Belongs to Group:	Select Output Copycode
Format/Length:	A8
Default Value:	(All objects of the selected library and type)

Press PF8 **Next** to continue.

Show Generation Results

After the generation is complete, the generation results summary is displayed.

```
13:51:11               ***** NATURAL XML Toolkit *****               2007-01-19
               - Generate from Document Type Definition -       Library SYSEXXT

 Generation Results
 Generate for DTD/ino schema
   File .....: /nat_64/proj/natc/42/samples/sysexxt/empl.dtd
   Read DTD/Tamino Schema done.
 Parser (Callback) Copycode
   Library ..: SYSEXXT
   Source ...: A1
   Parse Copycode generation done.
 Serialize (Compress XML) Copycode
   Library ..: SYSEXXT
   Source ...: A2
   Serialize Copycode generation done.
 Data Area
   Library ..: SYSEXXT
   Source ...: A3
   Data Area Generation done.

Generation done.
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                    Prev        Finis             Canc
```

Field Descriptions

Summary

Format/Length:	A253/1:v

Press PF9 **Finis**(h) to end the generation process.

74 Natural Simple XML Parser

The following topics are covered:

Parser Description and Example

The Natural simple XML Parser enables you to parse XML documents with standard Natural programs. The parser sends an event, or runs an internal subroutine callback if the next part of the document is parsed. The inline subroutine "CALLBACK" is called with the name of the current element, text, comment within an xpath-like syntax. The parser engine is included as copy code "PARSER_X". If an error occurs during parsing, e.g. the document is not wellformed, the "PARSER_ERROR" inline subroutine is called and then the parser is canceled with "ESCAPE SUBROUTINE" (see also **Parser Restrictions**).

For extended error handling, it is possible to change the operand6 "Error Message Text" and operand7 "Error Number" to a value less than or equal to -9000. Then the "PARSER_ERROR" inline subroutine is called and the (sub)program is canceled with "ESCAPE SUBROUTINE". If other values are less than or equal to -8000, only the parser is canceled with "ESCAPE SUBROUTINE".

The major variables of the parser are defined at the Local Data Area "PARSER-X".

The parser copycode takes the following operands:

Operand	Format/Length	Description
1	A	XML file to be parsed
2	A	**ex-XPATH** to repesent element structure
3	A1	Type of the XPATH content: ? Processing instruction D DOCTYPE ! Comment C CDATA section T Starting Tag @ Attribute / Close Tag
4	A	Parsed Data
5	L	Is TRUE if Parsed Data is empty
6	A	Error Message Text
7	I4	Error Number

Return value of the XPATH data:

ex-Xpath	XML Structure
?	<? ... ?>
!DOCTYPE	<!DOCTYPE ... >
!DOCTYPE[	<!DOCTYPE .. [...]>
![CDATA[	<![CDATA[...]]>
!--	<!-- -->
!	<! .. >
doc	<doc>
doc doc/foo doc/foo/$ doc/foo// doc//	<doc><foo>text</foo></doc>
doc doc/@a1 doc//	<doc a1="a" />
doc doc/@a1 doc/@a2 doc/$ doc//	<doc a1="a" a2="b">text</doc>
doc doc/$ doc/foo doc/foo/$ doc/foo// doc/$ doc//	<doc> <foo>text</foo> </doc>
doc doc/![CDATA[doc//	<doc><![CDATA[...]]></doc>
doc doc/!-- doc//	<doc><!-- ... --></doc>

Program Example:

```
* ------------------------------------------------------------------------
* CLASS  NATURAL XML TOOLKIT - UTILITIES
*
*         PARSER
*
* DESCRIPTION
*               Parse given XML
*
*
* AUTHOR        SAG   01.2006
*
* VERSION       6.2.
*
* (c) Copyright Software AG 2006. All rights reserved.
*
* ------------------------------------------------------------------------
*
DEFINE DATA LOCAL
1 XML_PARSER_INPUT             (A) DYNAMIC
1 XML_PARSER_ERROR_TEXT        (A253)
1 XML_PARSER_RESPONSE          (I4)
LOCAL USING PARSER-X           /* parser internal data - do not change
LOCAL
1 XML_PARSER_XPATH             (A) DYNAMIC
1 XML_PARSER_XPATH_TYPE        (A1)
1 XML_PARSER_CONTENT           (A) DYNAMIC
1 XML_PARSER_CONTENT_IS_EMPTY  (L)
*
1 ANFANG                       (T)
* OUT                          (A) DYNAMIC
1 OUT                          (A126)
*
END-DEFINE
*
FORMAT (0) LS=128 PS=40
*
DEFINE WORK FILE 12 "E:\EMPLOYEE1.XML" TYPE "UNFORMATTED"
READ WORK FILE 12 XML_PARSER_INPUT
END-WORK
CLOSE WORK FILE 12
*
*
* ------------------------------------------------- INCLUDE THE PARSER
INCLUDE PARSER_X 'XML_PARSER_INPUT' /* XML file to be parsed
  'XML_PARSER_XPATH'                /* XPATH to represent element...
  'XML_PARSER_XPATH_TYPE'           /* Type of callback
  'XML_PARSER_CONTENT'              /* Content of element found
  'XML_PARSER_CONTENT_IS_EMPTY'     /* Is TRUE if element is empty
```

```
  'XML_PARSER_ERROR_TEXT'           /* error Message
  'XML_PARSER_RESPONSE'             /* Error NR; 0 = OK
*
*
DEFINE SUBROUTINE CALLBACK
IF XML_PARSER_CONTENT_IS_EMPTY THEN
  IF XML_PARSER_XPATH_TYPE NE "T" AND XML_PARSER_XPATH_TYPE NE "/" THEN
    COMPRESS XML_PARSER_XPATH "(NULL)" INTO OUT WITH DELIMITER "="
  ELSE
    OUT := XML_PARSER_XPATH
  END-IF
ELSE
  COMPRESS XML_PARSER_XPATH XML_PARSER_CONTENT INTO OUT WITH DELIMITER "="
END-IF
WRITE OUT
END-SUBROUTINE
/*
DEFINE SUBROUTINE PARSER_ERROR
OUT := XML_PARSER_ERROR_TEXT
WRITE OUT
END-SUBROUTINE
END
```

With a given result document from Tamino for the Employee data, the result of this program looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Employee xmlns:ino="http://namespaces.softwareag.com/tamino/response2" ino:id="560"
Personnel-ID="20006900">
<Full-Name>
<First-Name>JOE</First-Name>
<Name>ATHERTON</Name>
</Full-Name>
<Mar-Stat>S</Mar-Stat>
<Sex>M</Sex>
<Birth>1941-02-21</Birth>
<Full-Address>
<Address-Line>11603 HUNTERS GREEN</Address-Line>
<Address-Line>SYRACUSE</Address-Line>
<Address-Line>NY</Address-Line>
<City>SYRACUSE</City>
<Zip>13201</Zip>
<Post-Code>13201</Post-Code>
<Country>USA</Country>
</Full-Address>
<Telephone>
<Phone>173-9859</Phone>
<Area-Code>315</Area-Code>
</Telephone>
<Dept>TECH10</Dept>
```

```
<Job-Title>ANALYST</Job-Title>
<Income>
<Curr-Code>USD</Curr-Code>
<Salary>43000</Salary>
</Income>
<Income>
<Curr-Code>USD</Curr-Code>
<Salary>39500</Salary>
</Income>
<Income>
<Curr-Code>USD</Curr-Code>
<Salary>36700</Salary>
</Income>
<Income>
<Curr-Code>USD</Curr-Code>
<Salary>34400</Salary>
</Income>
<Income>
<Curr-Code>USD</Curr-Code>
<Salary>32600</Salary>
</Income>
<Leave-Data>
<Leave-Due>19</Leave-Due>
<Leave-Taken>4</Leave-Taken>
</Leave-Data>
<Leave-Booked>
<Leave-Start>19980112</Leave-Start>
<Leave-End>19980112</Leave-End>
</Leave-Booked>
<Leave-Booked>
<Leave-Start>19980605</Leave-Start>
<Leave-End>19980605</Leave-End>
</Leave-Booked>
<Leave-Booked>
<Leave-Start>19980916</Leave-Start>
<Leave-End>19980916</Leave-End>
</Leave-Booked>
<Lang>ENG</Lang>
</Employee>
```

Note: There is no line break in the whole document.

The result of the above Natural program looks like this:

```
?=xml version="1.0" encoding="ISO-8859-1"
Employee
Employee/@xmlns:ino=http://namespaces.softwareag.com/tamino/response2
Employee/@ino:id=560
Employee/@Personnel-ID=20006900
Employee/Full-Name
Employee/Full-Name/First-Name
Employee/Full-Name/First-Name/$=JOE
Employee/Full-Name/First-Name//
Employee/Full-Name/Name
Employee/Full-Name/Name/$=ATHERTON
Employee/Full-Name/Name//
Employee/Full-Name//
Employee/Mar-Stat
Employee/Mar-Stat/$=S
Employee/Mar-Stat//
Employee/Sex
Employee/Sex/$=M
Employee/Sex//
Employee/Birth
Employee/Birth/$=1941-02-21
Employee/Birth//
Employee/Full-Address
Employee/Full-Address/Address-Line
Employee/Full-Address/Address-Line/$=11603 HUNTERS GREEN
Employee/Full-Address/Address-Line//
Employee/Full-Address/Address-Line
Employee/Full-Address/Address-Line/$=SYRACUSE
Employee/Full-Address/Address-Line//
Employee/Full-Address/Address-Line
Employee/Full-Address/Address-Line/$=NY
Employee/Full-Address/Address-Line//
Employee/Full-Address/City
Employee/Full-Address/City/$=SYRACUSE
Employee/Full-Address/City//
Employee/Full-Address/Zip
Employee/Full-Address/Zip/$=13201
Employee/Full-Address/Zip//
Employee/Full-Address/Post-Code
Employee/Full-Address/Post-Code/$=13201
Employee/Full-Address/Post-Code//
Employee/Full-Address/Country
Employee/Full-Address/Country/$=USA
Employee/Full-Address/Country//
Employee/Full-Address//
Employee/Telephone
Employee/Telephone/Phone
Employee/Telephone/Phone/$=173-9859
```

```
Employee/Telephone/Phone//
Employee/Telephone/Area-Code
Employee/Telephone/Area-Code/$=315
Employee/Telephone/Area-Code//
Employee/Telephone//
Employee/Dept
Employee/Dept/$=TECH10
Employee/Dept//
Employee/Job-Title
Employee/Job-Title/$=ANALYST
Employee/Job-Title//
Employee/Income
Employee/Income/Curr-Code
Employee/Income/Curr-Code/$=USD
Employee/Income/Curr-Code//
Employee/Income/Salary
Employee/Income/Salary/$=43000
Employee/Income/Salary//
Employee/Income//
Employee/Income
Employee/Income/Curr-Code
Employee/Income/Curr-Code/$=USD
Employee/Income/Curr-Code//
Employee/Income/Salary
Employee/Income/Salary/$=39500
Employee/Income/Salary//
Employee/Income//
Employee/Income
Employee/Income/Curr-Code
Employee/Income/Curr-Code/$=USD
Employee/Income/Curr-Code//
Employee/Income/Salary
Employee/Income/Salary/$=36700
Employee/Income/Salary//
Employee/Income//
Employee/Income
Employee/Income/Curr-Code
Employee/Income/Curr-Code/$=USD
Employee/Income/Curr-Code//
Employee/Income/Salary
Employee/Income/Salary/$=34400
Employee/Income/Salary//
Employee/Income//
Employee/Income
Employee/Income/Curr-Code
Employee/Income/Curr-Code/$=USD
Employee/Income/Curr-Code//
Employee/Income/Salary
Employee/Income/Salary/$=32600
Employee/Income/Salary//
Employee/Income//
Employee/Leave-Data
```

```
Employee/Leave-Data/Leave-Due
Employee/Leave-Data/Leave-Due/$=19
Employee/Leave-Data/Leave-Due//
Employee/Leave-Data/Leave-Taken
Employee/Leave-Data/Leave-Taken/$=4
Employee/Leave-Data/Leave-Taken//
Employee/Leave-Data//
Employee/Leave-Booked
Employee/Leave-Booked/Leave-Start
Employee/Leave-Booked/Leave-Start/$=19980112
Employee/Leave-Booked/Leave-Start//
Employee/Leave-Booked/Leave-End
Employee/Leave-Booked/Leave-End/$=19980112
Employee/Leave-Booked/Leave-End//
Employee/Leave-Booked//
Employee/Leave-Booked
Employee/Leave-Booked/Leave-Start
Employee/Leave-Booked/Leave-Start/$=19980605
Employee/Leave-Booked/Leave-Start//
Employee/Leave-Booked/Leave-End
Employee/Leave-Booked/Leave-End/$=19980605
Employee/Leave-Booked/Leave-End//
Employee/Leave-Booked//
Employee/Leave-Booked
Employee/Leave-Booked/Leave-Start
Employee/Leave-Booked/Leave-Start/$=19980916
Employee/Leave-Booked/Leave-Start//
Employee/Leave-Booked/Leave-End
Employee/Leave-Booked/Leave-End/$=19980916
Employee/Leave-Booked/Leave-End//
Employee/Leave-Booked//
Employee/Lang
Employee/Lang/$=ENG
Employee/Lang//
Employee//
```

Parser Restrictions

The parser does not handle:

- Composition of a tag (incl. processing instruction). Only start-tag must be equal to end-tag (incl. processing instruction).

Example:

```
<.doc></.doc> <!-- invalid character in tag -->
<doc><? ?></doc> <!-- invalid whitespace -->
<doc>&#RE;</doc> <!-- invalid character in tag -->
```

- Character or entity references

  Example:

```
<doc>& no refc</doc> <!-- missing semicolon --> <doc a1=v1></doc>
<!-- string literal expected -->
```

- Exact handling of CDATA-Sections

  Example:

```
<doc><![CDATA [ stuff]]></doc> <!-- must be CDATA[ -->
```

- Content of an entity/processing instruction

  Example:

```
<doc>]]></doc> <!-- ]] not allowed -->
```

- Number of tags/attributes
- Headerinformation
- Unicode-charset (supports ISO-8859-1)

75 Examples

The following examples are included:

Serialize Copycode

Using the XML Toolkit, a copycode can be generated that can be used to convert a Natural group structure into an XML document.

The callback copycode takes the following operands:

Operand	Format/Length	Description	from PARSER-X
1	A	ex-XPATH to represent element structure	operand2
2	A1	Type of the XPATH content: ? Processing instruction D DOCTYPE ! Comment C CDATA section T Starting Tag @ Attribute / Close Tag	operand3
3	A	Parsed Data	operand4
4	L	Is TRUE if Parsed Data is empty	operand5
5	I4	Counter Variable 1st Dimension	
6	I4	Counter Variable 2nd Dimension	
7	I4	Counter Variable 3rd Dimension	

Copycode Example EMPL-C:

```
*
------------------------------------------------------------------------ * Parameter
Definition * * &1& 'XML' /* XML Document * &2& '#CX' /* Counter
Variable 1st Dimension * &3& '#CY' /* Counter Variable 2nd Dimension *
&4& '#CZ' /* Counter Variable 3rd Dimension *
------------------------------------------------------------------------
* DTD E-\SAG\nat\NATAPPS\FUSER\XMLTK\RES\empl.dtd COMPRESS &1& '<EMPLOYEE'
' PERSONNEL-ID="'EMPLOYEE.PERSONNEL-ID '"' '>' INTO &1& LEAVING NO
/* now the children COMPRESS &1& '<FULL-NAME' '>' INTO &1&
LEAVING NO /* now the children COMPRESS &1& '<FIRST-NAME' '>' EMPLOYEE.FIRST-NAME
'</FIRST-NAME>' INTO &1& LEAVING NO COMPRESS &1& '<NAME'
'>' EMPLOYEE.NAME '</NAME>' INTO &1& LEAVING NO /* COMPRESS &1&
'</FULL-NAME>' INTO &1& LEAVING NO COMPRESS &1& '<FULL-ADDRESS'
```

```
'>' INTO &1& LEAVING NO /* now the children FOR &2& = 1 TO
EMPLOYEE.C@ADDRESS-LINE COMPRESS &1& '<ADDRESS-LINE' '>' EMPLOYEE.ADDRESS-LINE(&2&)
'</ADDRESS-LINE>' INTO &1& LEAVING NO END-FOR COMPRESS &1&
'<CITY' '>' EMPLOYEE.CITY '</CITY>' INTO &1& LEAVING NO COMPRESS
&1& '<ZIP' '>' EMPLOYEE.ZIP '</ZIP>' INTO &1& LEAVING
NO COMPRESS &1& '<COUNTRY' '>' EMPLOYEE.COUNTRY '</COUNTRY>'
INTO &1& LEAVING NO /* COMPRESS &1& '</FULL-ADDRESS>' INTO
&1& LEAVING NO COMPRESS &1& '<TELEPHONE' '>' INTO &1&
LEAVING NO /* now the children COMPRESS &1& '<PHONE' '>' EMPLOYEE.PHONE
'</PHONE>' INTO &1& LEAVING NO COMPRESS &1& '<AREA-CODE'
'>' EMPLOYEE.AREA-CODE '</AREA-CODE>' INTO &1& LEAVING NO /*
COMPRESS &1& '</TELEPHONE>' INTO &1& LEAVING NO COMPRESS
&1& '<JOB-TITLE' '>' EMPLOYEE.JOB-TITLE '</JOB-TITLE>' INTO
&1& LEAVING NO FOR &2& = 1 TO EMPLOYEE.C@INCOME COMPRESS &1&
'<INCOME' '>' INTO &1& LEAVING NO /* now the children COMPRESS &1&
'<SALARY' '>' EMPLOYEE.SALARY(&2&) '</SALARY>' INTO &1&
LEAVING NO FOR &3& = 1 TO EMPLOYEE.C@BONUS(&2&) COMPRESS &1&
'<BONUS' '>' EMPLOYEE.BONUS(&2&,&3&) '</BONUS>' INTO
&1& LEAVING NO END-FOR /* COMPRESS &1& '</INCOME>' INTO
&1& LEAVING NO END-FOR /* COMPRESS &1& '</EMPLOYEE>' INTO
&1& LEAVING NO
```

Program Example:

```
*
---------------------------------------------------------------------- * CLASS
NATURAL XML TOOLKIT * * * DESCRIPTION * Serialize a given Data structure. * *
* AUTHOR SAG 01.2006 * * VERSION 6.2. * * (c) Copyright Software AG 2006. All
rights reserved. * *
----------------------------------------------------------------------
* DEFINE DATA
LOCAL USING EMPL /* add generated data structure LOCAL 1
XML (A) DYNAMIC * 1 OUT (A72) 1 II (I4) * 1 OUTDYN (A) DYNAMIC 1 OBJLEN (I4) 1
OBJEND (I4) 1 OBJSTART (I4) 1 OBJLINE (I4) * 1 #CX (I4) 1 #CY (I4) 1 #CZ (I4)
END-DEFINE * EMPLOYEE.PERSONNEL-ID := 4711 * EMPLOYEE.FIRST-NAME := "ADKINSON"
EMPLOYEE.NAME := "MARTHA" * EMPLOYEE.C@ADDRESS-LINE := 2 EMPLOYEE.ADDRESS-LINE(1)
:= "8603 GARLAND COURT" EMPLOYEE.ADDRESS-LINE(2) := "FRAMINGHAM"
EMPLOYEE.ADDRESS-LINE(2)
:= "MA" EMPLOYEE.CITY := "FRAMINGHAM" EMPLOYEE.ZIP := "17010" EMPLOYEE.COUNTRY
:= "USA" * EMPLOYEE.AREA-CODE := "617" EMPLOYEE.PHONE := "210-4703" *
EMPLOYEE.JOB-TITLE
:= "MANAGER" EMPLOYEE.C@INCOME := 2 EMPLOYEE.SALARY(1) := 47000 EMPLOYEE.C@BONUS(1)
:= 2 EMPLOYEE.BONUS(1,1) := 10500 EMPLOYEE.BONUS(1,2) := 7875 * EMPLOYEE.SALARY(2)
:= 47000 EMPLOYEE.C@BONUS(2) := 1 EMPLOYEE.BONUS(2,1) := 35700 *
INCLUDE EMPL-C
"XML" "#CX" "#CY" "#CZ" /* add generated Serialize * FOR II = 1 TO *LENGTH(XML)
STEP 72 OUT := SUBSTR(XML,II) WRITE OUT END-FOR * NEWPAGE * /* WRITE COMPLETE
(A) DYNAMIC VARIABLE IF POSSIBLE USE CR AND IGNORE LF OBJSTART := 1 * EXAMINE
xml FOR "><" REPLACE WITH ">" - H'0A' -"<" EXAMINE xml FOR H'0A' GIVING
POSITION OBJEND * REPEAT WHILE OBJEND NE 0 /* IF OBJSTART GT 0 THEN ADD OBJSTART
TO OBJEND END-IF /* OBJLEN := OBJEND - OBJSTART -1 /* IF OBJLEN > 0 THEN OUTDYN
```

```
:= SUBSTRING(xml, OBJSTART, OBJLEN) /* FOR OBJLINE = 1 TO *LENGTH(OUTDYN) STEP
72 OUT := SUBSTR (OUTDYN,OBJLINE) WRITE OUT END-FOR ELSE WRITE " " END-IF /* OBJSTART
:= OBJEND IF OBJSTART GT *LENGTH(xml) ESCAPE BOTTOM END-IF /* EXAMINE
SUBSTRING(xml,OBJSTART)
FOR H'0A' GIVING POSITION OBJEND END-REPEAT * END
```

Natural PDA EMPL Used:

```
DEFINE DATA PARAMETER 1 EMPLOYEE 2 ATTRIBUTES_OF_EMPLOYEE
3 PERSONNEL-ID(A8) * 2 FULL-NAME 3 FIRST-NAME(A20) 3 NAME(A20) * 2 FULL-ADDRESS
3 C@ADDRESS-LINE(I4) 3 ADDRESS-LINE(A20/1:6) 3 CITY(A20) 3 ZIP(A20) 3 COUNTRY(A3)
* 2 TELEPHONE 3 AREA-CODE(A6) 3 PHONE(A15) * 2 JOB-TITLE(A25) * 2 C@INCOME(I4)
2 INCOME(1:6) 3 SALARY(A9) 3 C@BONUS(I4) 3 BONUS(A9/1:4) END-DEFINE
```

Generated Natural Data Area

Using the XML Toolkit, a Natural Data Area, or more precisely a Local Data Area, Parameter Data Area or Global Data Area, can be generated that represents a given Document Type Definition.

Generation Rules:

- Each Empty Element without Attributes (<!ELEMENT br EMPTY>) is generated as a Natural variable of Type B1. This is necessary, because empty Natural groups are not allowed.
- Each Empty Element with Attributes (<!ELEMENT br EMPTY><!ATTLIST br width CDATA #IMPLIED>) is generated as a Natural group.
- Each Element with content (<!ELEMENT b (#PCDATA)>)is generated as a Natural variable of type A253.
- Each Sequence of Elements (<!ELEMENT spec (front, body*, back?)>) or Choice of Elements (<!ELEMENT div1 (p | list | note)>) is generated as a Natural group.
- Each clasped Sequence or Choice (<!ELEMENT address ((street, housenumber), (zip, city))>) is generated as a special group with the name prefix "##PSEUDO". This gives the possibility to represent the context or possible multiplicities.
- Each Attribute (<!ATTLIST br width CDATA #IMPLIED>) of an Element is generated as variable of Type A253 belonging to a group with the name prefix "ATTRIBUTES_OF_" followed by the name of the element.
- Multiple Elements are always generated as arrays of Dimension 1:v. The upper bound of the generated array has to be changed manually.
- If an Element is defined multiple (<!ELEMENT spec (front, *body**)>), an additional counter field C@BODY, is generated to specify the number of available elements.

- All names used inside the DTD are converted into upper case, because Natural names are not case sensitive. Duplicate names inside a generated group will be a extended with an suffix to make the names unique.
- Special Characters not valid for Natural names are converted into valid Natural names. For the conversion settings, see the option dialog of the XML Toolkit.

Restrictions:

- Elements with Mixed content data (<!ELEMENT p (#PCDATA | a | ul | b | i | em)*>) are not supported.
- DTDs that result in Natural data structures can not be used within Natural, because Natural only supports data structures with a maximum of three dimensions.

Example DTD:

```
<!ELEMENT EMPLOYEE  (FULL-NAME , FULL-ADDRESS , TELEPHONE ,JOB-TITLE, INCOME* )>
<!ATTLIST EMPLOYEE PERSONNEL-ID CDATA  #REQUIRED >

<!ELEMENT FULL-NAME  (FIRST-NAME , NAME )>
<!ELEMENT FIRST-NAME  (#PCDATA )>
<!ELEMENT NAME  (#PCDATA )>

<!ELEMENT FULL-ADDRESS  (ADDRESS-LINE* , CITY , ZIP , COUNTRY )>
<!ELEMENT ADDRESS-LINE  (#PCDATA )>
<!ELEMENT CITY  (#PCDATA )>
<!ELEMENT ZIP  (#PCDATA )>
<!ELEMENT COUNTRY  (#PCDATA )>

<!ELEMENT TELEPHONE  (PHONE , AREA-CODE )>
<!ELEMENT PHONE  (#PCDATA )>
<!ELEMENT AREA-CODE  (#PCDATA )>

<!ELEMENT JOB-TITLE (#PCDATA )>

<!ELEMENT INCOME  (SALARY , BONUS* )>
<!ELEMENT SALARY  (#PCDATA )>
<!ELEMENT BONUS  (#PCDATA )>
```

Generated Natural Data Area (*italic* written parts of the DTD, but necessary for Natural):

```
DEFINE DATA PARAMETER
1 EMPLOYEE
  2 ATTRIBUTES_OF_EMPLOYEE
    3 PERSONNEL-ID(A253)
*
  2 FULL-NAME
    3 FIRST-NAME(A253)
    3 NAME(A253)
*
  2 FULL-ADDRESS
    3 C@ADDRESS-LINE(I4)
    3 ADDRESS-LINE(A253/1:v)
    3 CITY(A253)
    3 ZIP(A253)
    3 COUNTRY(A253)
*
  2 TELEPHONE
    3 AREA-CODE(A253)
    3 PHONE(A253)
*
  2 JOB-TITLE(A253)
*
  2 C@INCOME(I4)
  2 INCOME(1:v)
    3 SALARY(A253)
    3 C@BONUS(I4)
    3 BONUS(A253/1:v)
END-DEFINE
```

Natural DTD Parser

Translation Rules:

Natural	Document Type Definiton
1 G1 2 E1 (Aâ€¦)	<!ELEMENT G1 (E1)> <!ELEMENT E1 (#PCDATA)>
1 G1 2 E1 (Aâ€¦) 2 E2 (Aâ€¦) 2 E3 (Aâ€¦)	<!ELEMENT G1 (E1, E2, E3)> <!ELEMENT E1 (#PCDATA)> <!ELEMENT E2 (#PCDATA)> <!ELEMENT E3 (#PCDATA)>
1 C@E1_MAX (I4) CONST <10> 1 G1 2 C@E1 (I4)	<!ELEMENT G1 (E1*)> <!ELEMENT E1 (#PCDATA)>

Natural	Document Type Definiton
2 E1 (Aâ€¦/1:C@E1_MAX)	
1 C@E1_MAX (I4) CONST <10> 1 G1 2 C@E1 (I4) 2 E1 (Aâ€¦/1:C@E1_MAX)	<!ELEMENT G1 (E1+)> <!ELEMENT E1 (#PCDATA)>
1 G1 2 E1 (Aâ€¦)	<!ELEMENT G1 (E1?)> <!ELEMENT E1 (#PCDATA)>
1 G1 2 E1 (Aâ€¦) 2 E2 (Aâ€¦) 2 E3 (Aâ€¦)	<!ELEMENT G1 (E1| E2| E3)> <!ELEMENT E1 (#PCDATA)> <!ELEMENT E2 (#PCDATA)> <!ELEMENT E3 (#PCDATA)>
1 G1 2 E1 (Aâ€¦) 2 E2 (Aâ€¦) 2 G2 2 E1_2 (Aâ€¦) 2 E3 (Aâ€¦)	<!ELEMENT G1 (E1, E2, G2)> <!ELEMENT E1 (#PCDATA)> <!ELEMENT E2 (#PCDATA)> <!ELEMENT G2 (E1, E3)> <!ELEMENT E3 (#PCDATA)>
1 #G1 2 #E1 (Aâ€¦)	<!ELEMENT G1 (E1)> <!ELEMENT E1 (#PCDATA)>
2 E1 (Aâ€¦) 3 ATTRIBUTES_OF_E1 4 A1 (Aâ€¦) CONST <'schema'> 4 A2 (Aâ€¦) 4 A3 (Aâ€¦)	<!ELEMENT E1 (#PCDATA)> <!ATTLIST E1 A1 #FIXED "schema" A2 NMTOKEN #IMPLIED A3 ID #REQUIRED>

Generated Type Definition

Using the XML Toolkit, a Natural Data Area, or more precisely a Local Data Area, Parameter Data Area or Global Data Area, can be used to generate a Document Type Definition.

Generation Rules:

- A Natural variable will result in an element with content.
- A Natural group will result in a sequence of elements.
- Multiple variables or groups will be generated with multiplicity "zero or more".
- Special characters not valid for XML names are converted into valid names. For the conversion settings, see the **options screen** of the XML Toolkit.

Example Natural Data Area:

```
DEFINE DATA LOCAL
1 NAT$EMPLOYEE
  2 ATTRIBUTES_OF_NAT$EMPLOYEE
    3 PERSONNEL/ID(A8)
  2 C@MAN@WORK(I4)
  2 MAN@WORK
    3 JOB(A10)
  2 A$TEST$MAKL(I4)
  2 AS/FA/SD(P7.5)
  2 #ASDFAS(F4)
  2 ASF#AS(N9)
  2 A-SF-D(A) Dynamic
  2 INC@OME(1:6)
    3 C@BONUS(I4)
    3 BONUS(A9/1:4)
END-DEFINE
```

Generated DTD:

```
<!-- DTD XMLTOOLS BEISP -->
<!ELEMENT NATdollarEMPLOYEE ( MANatWORK , AdollarTESTdollarMAKL ,
           ASslashFAslashSD , hashASDFAS , ASFhashAS , A-SF-D , INCatOME* ) >
<!ATTLIST NATdollarEMPLOYEE PERSONNELslashID CDATA #IMPLIED >
<!ELEMENT MANatWORK ( JOB ) >
<!ELEMENT JOB (#PCDATA) >
<!ELEMENT AdollarTESTdollarMAKL (#PCDATA) >
<!ELEMENT ASslashFAslashSD (#PCDATA) >
<!ELEMENT hashASDFAS (#PCDATA) >
<!ELEMENT ASFhashAS (#PCDATA) >
<!ELEMENT A-SF-D (#PCDATA) >
<!ELEMENT INCatOME ( BONUS* ) >
<!ELEMENT BONUS (#PCDATA) >
```

Parser CALLBACK Copycode

Using the XML Toolkit, a copycode can be generated that can be used with the **N**atural S **i**mple **X**ML Pars **e**r.

The callback copycode takes the following operands:

Operand	Format/Length	Description	from PARSER-X
1	A	**ex-XPATH** to repesent element structure	operand2
2	A1	Type of the XPATH content: ? Processing instruction D DOCTYPE ! Comment C CDATA section T Starting Tag @ Attribute / Close Tag	operand3
3	A	Content of found element	operand4
4	L	Is TRUE if Parsed Data is empty	operand5
5	I4	Counter Variable 1stDimension	
6	I4	Counter Variable 2nd Dimension	
7	I4	Counter Variable 3rd Dimension	

Copycode Example EMPL-P:

```
* ------------------------------------------------------------------------
* Paremeter Definition
*
* &1& 'XML_PARSER_XPATH'              /* XPATH to repesent element...
* &2& 'XML_PARSER_XPATH_TYPE'         /* Type of the XPATH:
*                                        ? Processing instruction
*                                        D DOCTYPE
*                                        ! Comment
*                                        C CDATA section
*                                        T Starting Tag
*                                        @ Attribute
*                                        / Close Tag
*                                        $ Parsed Data
* &3& 'XML_PARSER_CONTENT'            /* Content of found element
* &4& 'XML_PARSER_CONTENT_IS_EMPTY'   /* Is TRUE if Content is empty
* &5& '#CX'                           /* Counter Variable 1st Dimension
* &6& '#CY'                           /* Counter Variable 2nd Dimension
* &7& '#CZ'                           /* Counter Variable 3rd Dimension
* ------------------------------------------------------------------------
*
DECIDE ON FIRST &1&
VALUE 'EMPLOYEE'
RESET EMPLOYEE
```

```
VALUE 'EMPLOYEE/@PERSONNEL-ID'
  /* #REQUIRED
  EMPLOYEE.PERSONNEL-ID := &3&
VALUE 'EMPLOYEE/FULL-NAME'
  IGNORE
VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME'
  IGNORE
VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME/$'
  EMPLOYEE.FIRST-NAME := &3&
VALUE 'EMPLOYEE/FULL-NAME/NAME'
  IGNORE
VALUE 'EMPLOYEE/FULL-NAME/NAME/$'
  EMPLOYEE.NAME := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/ADDRESS-LINE'
  /* OPTIONAL MULTIPLE IST: 18 PARENT: FULL-ADDRESS
  ADD 1 TO EMPLOYEE.C@ADDRESS-LINE
VALUE 'EMPLOYEE/FULL-ADDRESS/ADDRESS-LINE/$'
  &5& := EMPLOYEE.C@ADDRESS-LINE
  EMPLOYEE.ADDRESS-LINE(&5&) := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS/CITY'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/CITY/$'
  EMPLOYEE.CITY := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS/ZIP'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/ZIP/$'
  EMPLOYEE.ZIP := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS/COUNTRY'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/COUNTRY/$'
  EMPLOYEE.COUNTRY := &3&
VALUE 'EMPLOYEE/TELEPHONE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/PHONE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/PHONE/$'
  EMPLOYEE.PHONE := &3&
VALUE 'EMPLOYEE/TELEPHONE/AREA-CODE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/AREA-CODE/$'
  EMPLOYEE.AREA-CODE := &3&
VALUE 'EMPLOYEE/JOB-TITLE'
  IGNORE
VALUE 'EMPLOYEE/JOB-TITLE/$'
  EMPLOYEE.JOB-TITLE := &3&
VALUE 'EMPLOYEE/INCOME'
  /* OPTIONAL MULTIPLE IST: 18 PARENT: EMPLOYEE
  ADD 1 TO EMPLOYEE.C@INCOME
VALUE 'EMPLOYEE/INCOME/SALARY'
  IGNORE
```

```
VALUE 'EMPLOYEE/INCOME/SALARY/$'
  &5& := EMPLOYEE.C@INCOME
  EMPLOYEE.SALARY(&5&) := &3&
VALUE 'EMPLOYEE/INCOME/BONUS'
  /* OPTIONAL MULTIPLE IST: 18 PARENT: INCOME
  &5& := EMPLOYEE.C@INCOME
  ADD 1 TO EMPLOYEE.C@BONUS(&5&)
VALUE 'EMPLOYEE/INCOME/BONUS/$'
  &5& := EMPLOYEE.C@INCOME
  &6& := EMPLOYEE.C@BONUS(&5&)
  EMPLOYEE.BONUS(&5&,&6&) := &3&
NONE
IGNORE
END-DECIDE
```

Subprogram Example:

```
* ------------------------------------------------------------------------
* CLASS  NATURAL XML TOOLKIT - UTILITIES
*
*
* DESCRIPTION
*               Parse a given XML document.
*
*
* AUTHOR        SAG   01.2006
*
* VERSION       6.2.
*
* (c) Copyright Software AG 2006. All rights reserved.
*
* ------------------------------------------------------------------------
*
DEFINE DATA PARAMETER
1 XML_PARSER_INPUT             (A) DYNAMIC
PARAMETER USING EMPL
PARAMETER
1 XML_PARSER_ERROR_TEXT        (A253)
1 XML_PARSER_RESPONSE          (I2)
*
LOCAL USING PARSER-X
LOCAL
1 XML_PARSER_XPATH             (A) DYNAMIC
1 XML_PARSER_XPATH_TYPE        (A1)
1 XML_PARSER_CONTENT           (A) DYNAMIC
1 XML_PARSER_CONTENT_IS_EMPTY  (L)
*
LOCAL
1 #CX                          (I4)
1 #CY                          (I4)
1 #CZ                          (I4)
```

```
END-DEFINE
*
* ------------------------------------------------ INCLUDE THE PARSER
INCLUDE PARSER_X 'XML_PARSER_INPUT' /* XML file to be parsed
  'XML_PARSER_XPATH'                /* XPATH to repesent element...
  'XML_PARSER_XPATH_TYPE'           /* Type of callback
  'XML_PARSER_CONTENT'              /* Content of founld element
  'XML_PARSER_CONTENT_IS_EMPTY'     /* Is TRUE if element is empty
  'XML_PARSER_ERROR_TEXT'           /* error Message
  'XML_PARSER_RESPONSE'             /* Error NR; 0 = OK
*
* ------------------------------------------------ CALLBACK HANDLER
DEFINE SUBROUTINE CALLBACK
*
INCLUDE EMPL-P 'XML_PARSER_XPATH'   /* XPATH to repesent element...
  'XML_PARSER_XPATH_TYPE'           /* Type of callback
  'XML_PARSER_CONTENT'              /* Content of founld element
  'XML_PARSER_CONTENT_IS_EMPTY'     /* Is TRUE if element is empty
  '#CX'
  '#CY'
  '#CZ'
*
END-SUBROUTINE
/*
DEFINE SUBROUTINE PARSER_ERROR
IGNORE
END-SUBROUTINE
END
```

Natural PDA EMPL Used:

```

DEFINE DATA PARAMETER
1 EMPLOYEE
  2 ATTRIBUTES_OF_EMPLOYEE
    3 PERSONNEL-ID(A8)
*
  2 FULL-NAME
    3 FIRST-NAME(A20)
    3 NAME(A20)
*
  2 FULL-ADDRESS
    3 C@ADDRESS-LINE(I4)
    3 ADDRESS-LINE(A20/1:6)
    3 CITY(A20)
    3 ZIP(A20)
    3 COUNTRY(A3)
*
  2 TELEPHONE
    3 AREA-CODE(A6)
    3 PHONE(A15)
*
```

```
  2 JOB-TITLE(A25)
*
  2 C@INCOME(I4)
  2 INCOME(1:6)
    3 SALARY(A9)
    3 C@BONUS(I4)
    3 BONUS(A9/1:4)
END-DEFINE
```

76 Parser Error Messages

The following error messages will be produced by the parser:

Response	Error Text	Example
00	Parse ended without errors.	valid/*
-01	Wrong character set/Document does not start with '<'.	not-wf/sa/147.xml
-02	Processing instruction was not closed. Position %2%.	not-wf/sa/004.xml
-03	A CDATA section was not closed. Position %2%.	esi/001.xml
-04	!DOCTYPE section was not closed. Position %2%.	not-wf/sa/055.xml
-05	Incorrect syntax was used in a comment. Position %2%.	not-wf/sa/006.xml
-06	A comment was not closed. Position %2%.	not-wf/sa/027.xml
-07	A CDATA section was not closed. Position %2%	not-wf/sa/017.xml
-08	A comment section was not closed. Position %2%.	esi/002.xml
-09	Closing tag name was started with an invalid character. Position %2%.	not-wf/sa/019.xml
-10	Closing tag without starting element. Position %2%.	not-wf/sa/042.xml
-11	Closing tag '%3%' does not match the start tag '%1%'. Position %2%.	not-wf/sa/039.xml
-12	Closing tag was not closed. Position %2%.	esi/003.xml
-13	Closing tag '%1%' was not closed. Position %2%.	no example available
-14	Starting tag name was started with an invalid character. Position %2%.	not-wf/sa/035.xml
-15	Attribute name of tag '%1%' not found. Position %2%.	no example available
-16	Attribute name of tag '%1%' contains an invalid character. Position %2%.	not-wf/sa/001.xml
-17	Attribute value of tag '%1%' ending quotation mark missing. Position %2%.	not-wf/sa/013.xml
-18	Attribute value of tag '%1%' ending apostrophe missing. Position %2%.	esi/005.xml
-19	Starting tag section was not closed. Position %2%.	esi/006.xml
-20	Tag '%1%' section was not closed. Position %2%.	not-wf/sa/176.xml
-21	A section was not closed. Position %2%.	not-wf/sa/025.xml

Response	Error Text	Example
< -8000	User defined error messages, parser ends.	no example available
< -9000	User defined error messages, PARSER_ERROR is called and parser ends.	no example available

Stichwortverzeichnis