Natural

Web Technology

Version 9.3.4

October 2026

ADABAS & NATURAL

This document applies to Natural Version 9.3.4 and all subsequent releases.

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

Copyright © 1992-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: NATWIN-NNATWEBTECH-934-20261013

Table of Contents

Preface ... ix
1 About this Documentation ... 1
Document Conventions ... 2
Online Information and Support ... 2
Data Protection ... 3
I JSON Parser ... 5
2 JSON Parser ... 7
II Natural Web Interface ... 9
3 Introducing the Natural Web Interface ... 11
What is the Natural Web Interface ... 12
Architecture of the Natural Web Interface ... 12
Natural Web Interface Modules ... 15
Natural Web Interface Features ... 16
Functionality of the Natural Web Interface ... 17
Natural Web Interface Security ... 18
4 Natural Web Interface Installation and Configuration ... 19
5 Configuring the Natural Web Interface ... 21
Supported HTTP Servers ... 22
Configuring RPC and RPC Server ... 22
Configuring the DCOM Server ... 24
Configuring the Web Interface ... 24
Configuring the Web Interface for Apache 2.4.x (64bit) ... 28
Configuring an HTTP Server ... 29
Communication with Natural Security ... 29
6 Web Interface Troubleshooting ... 31
7 Natural Web Interface Essentials ... 33
8 Working with the Natural Web Interface ... 35
Setting up your Environment ... 36
Building Subprograms in Natural ... 37
III Natural Web Server Extensions ... 53
9 Natural Web Server Extensions - Introduction ... 55
About Natural Web Server Extensions ... 56
Installation - RPC / DCOM ... 56
Transformations ... 57
Variables ... 57
Error Logging and Messages ... 57
Calling Programs ... 58
10 Natural Web Server Extensions - Initialization File ... 59
About the Initialization File ... 60
RPC Parameters ... 60
PAL Parameters ... 61
DCOM Parameters ... 61
Natural Web Server Extension Settings ... 61

Data Transfer Settings ... 63
HTTP Server Variables ... 64
Additional Variables ... 65
Error Templates ... 65
11 Natural Web Server Extensions - Error Messages ... 69
12 Test Utility WEB-ONL3 with SYSWEB3 ... 73
Prerequisites ... 74
Running the Application ... 74
Supported Content Types ... 75
Input/Output Fields ... 76
13 Programming Tips ... 79
Editing in Lower Case ... 80
Quote vs. Apostrophe ... 80
Variables defined by Value ... 81
Access to Resources ... 81
Constant Values ... 81
Creating a New Page ... 82
DCOM / RPC ... 82
14 Web Interface Administration ... 83
Create a User-Defined Error Page ... 84
Create a User-Defined Error Page XML-Style ... 84
Alphanumeric-to-HTML Conversion ... 84
Alphanumeric-to-URL Conversion ... 85
15 Demonstration Application - without JavaScript ... 87
About the Demo Application without JavaScript ... 88
Design Decisions ... 89
Libraries, Modules, and Naming Conventions ... 89
Starting the Demonstration Application D3MENU ... 90
Starting the Natural Web Interface Online Manual ... 90
Requirements for the D3MENU Demonstration Application ... 90
16 Demonstration Application - with JavaScript ... 91
About the Demo Application with JavaScript ... 92
Design Decisions ... 93
Starting the Demonstration Application D4ENTER ... 93
Requirements ... 93
17 Natural Web Interface Error Messages ... 95
Error Messages ... 96
18 Migrate Natural Web Interface SYSWEB to SYSWEB3 ... 97
IV Web Interface Plug-In ... 99
19 Web Interface Plug-In ... 101
Before You Start ... 102
Invoking Web Interface Plug-In Commands ... 102
Web Interface Plug-In Functions ... 103
V Natural Web Online Documentation SYSWEB3 ... 127
20 Writes Binary to the Document ... 133

21 Clear Output Area 135
22 Set Document Content-Type 137
23 Count Size of Output Area 139
24 Generate Error Page 141
25 Writes to the Document and Converts to Valid HTML 143
26 Writes HTTP Settings to the Document 145
27 Info About Internal Values 147
28 End and Initialize Document 149
29 List All Environment Variables 151
30 Evaluate Mime-Type and Transfer/Data-Type 153
31 Evaluate Mime-Type and File Extension 155
32 Set Document Location 157
33 Read Environment Variable 159
34 Read Environment Variables Groups 161
35 Read Environment Text Area Variables 163
36 Write Text to Document 165
37 Write Newline to Output Area 167
38 Read Natural Source into X-Array of Dynamic 169
39 Text to HTML 171
40 Text to XML 173
41 Text to URL 175
42 Replace Inside Return Document 177
43 Load Resource 179
44 Read Resource 181
45 Write Resource 183
46 Check Resource 185
47 Delete Resource 187
48 Apply XSLT Processing on Return Document 189
49 Apply XSLT Processing on Return Document from Resource 191
50 Load Style Sheet from the Resource Directory and Apply XSLT Processing on Return Document 193
51 List Resource Files 195
52 Read Input Page 197
53 Read Output Page 199
54 Maintain an External Counter 201
55 Process User-Defined Tags 203
56 Anchor 207
57 Button 209
58 Checkbox Group 211
59 Comment Line 213
60 Level and Header 215
61 Image 217
62 Input 219
63 Line Break 221
64 Form 223

65 HTML Document 225
66 List 227
67 Paragraph 231
68 Radio Button Group 233
69 Horizontal Rule 235
70 Scrolling List 237
71 Table 239
72 Universal Tag 243
73 Text Area 245
74 Text to URL - Decoded 247
75 Time/Date String 249
76 List all Natural Libraries 251
77 Run Online Natural Web Interface Subprograms 253
78 Generate Natural Subprogram to use with Natural Web Interface 255
79 List All Data Passed From a HTTP Server to a Called Natural Subprogram 257
80 List Directory of a Natural Library 259
81 List Resources of a Natural Library 261
82 List All Parameters Passed From a HTTP Server To a Called Natural Subprogram 263
83 Return an HTML Page Saved as Natural Source Object 265
84 List the Current Natural Web Interface Settings 267
85 List Source of Natural Object 269
86 Online Documentation 271
87 List non-Natural File - Resource 273
VI XML Toolkit Plug-In 275
88 Introduction to the XML Toolkit Plug-In 277
About the XML Toolkit Plug-In 278
XML Toolkit Features 278
XML Toolkit Description 278
Considerations and Limitations 283
89 Using the XML Toolkit 291
Invoking the Application 292
Getting Help 292
90 Setting up Specific Generation Options 293
Invoking the Natural XML Options Menu 294
Generation 294
Path 297
Saving Your Options Permanently 299
91 Using a Natural Data Source 301
Select Natural Data Area 302
Select Data Type 304
Generate File with DTD Definition or XML Schema 305
Generate a serializer for an XML document 306
Generate a parser for an XML document 307

Parameter Settings 309
Select Root Group 309
Show Generation Report 310
92 Using an external Data Source 311
Generate from Document Type Definition or XML Schema 312
Select Root Element or Document Type 313
Select Recursion Level 314
Generate Natural Data Area 315
Generate Copycode for Serialization 317
Generate Subprogram for Serialization 318
Generate Copycode for XML Parser Callback 318
Generate Subprogram for XML Parser Callback 320
Show Generation Results 320
93 Natural Simple XML Parser 323
Parser Description and Example 324
Parser Restrictions 331
94 Examples 333
Serialize Copycode 334
Serialize Subroutine 336
Generated Natural Data Area 344
Natural DTD Parser 346
Generated Type Definition 347
Parser CALLBACK Copycode 348
95 XML Parser Error Messages 353

Preface

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

The following topics are covered:

JSON Parser	The PARSE JSON statement allows developers to parse JSON documents from within a Natural program.
Natural Web Interface	The Natural Web Interface is a link between a Web Server (HTTP server) and your Natural environment.
Web Interface Plug-In	The Web Interface plug-in is an optional plug-in unit for Natural Studio.
XML Toolkit Plug-In	The XML Toolkit plug-in enables developers to process XML documents within Natural.

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

For further details on the Natural statements that can be used together with Natural´s web technologies, please refer to Statements Grouped by Functions > Internet and XML in the *Statements documentation*.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

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

Online Information and Support

Product Documentation

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

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

I JSON Parser

2 JSON Parser

You can parse JSON documents from within a Natural program using the `PARSE JSON` statement.

For detailed information, see *PARSE JSON* in section *Statements* of the documentation.

II Natural Web Interface

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

The Natural Web Interface documentation comprises the following documents:

- **Introducing the Natural Web Interface**
- **Natural Web Interface Installation and Configuration**
- **Natural Web Interface Essentials**
- **Migrate Natural Web Interface SYSWEB to SYSWEB3**
- **Natural Web Online Documentation SYSWEB3**

3 Introducing the Natural Web Interface

- What is the Natural Web Interface 12
- Architecture of the Natural Web Interface 12
- Natural Web Interface Modules 15
- Natural Web Interface Features 16
- Functionality of the Natural Web Interface 17
- Natural Web Interface Security 18

Companies and organizations offer their information and services via the Internet. Static HTML pages are not sufficient for this task. 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 running operating systems such as z/OS.

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 z/OS) or on the same machine as the HTTP server (e.g. Apache).

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).

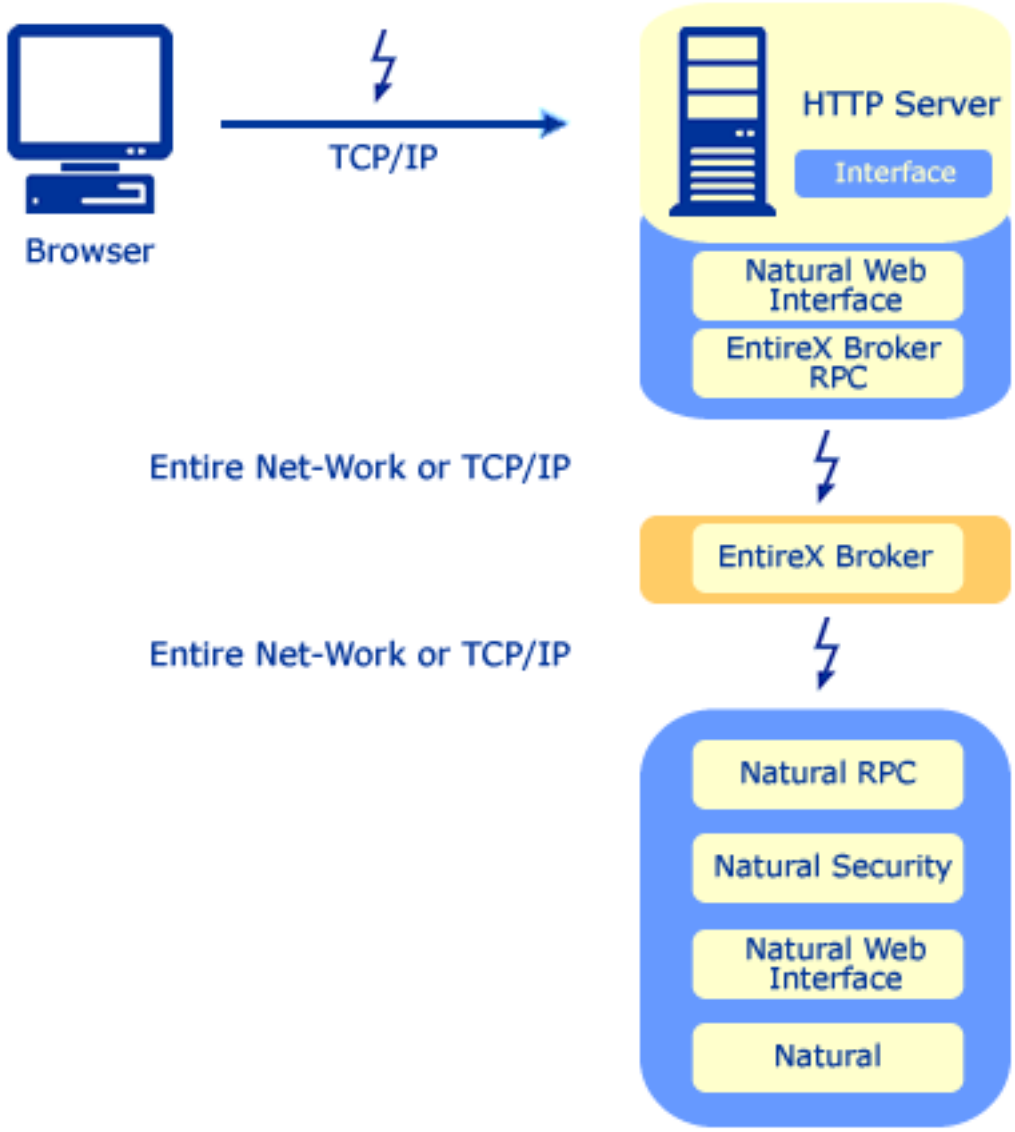
Architecture of the Natural Web Interface

The following topics are covered:

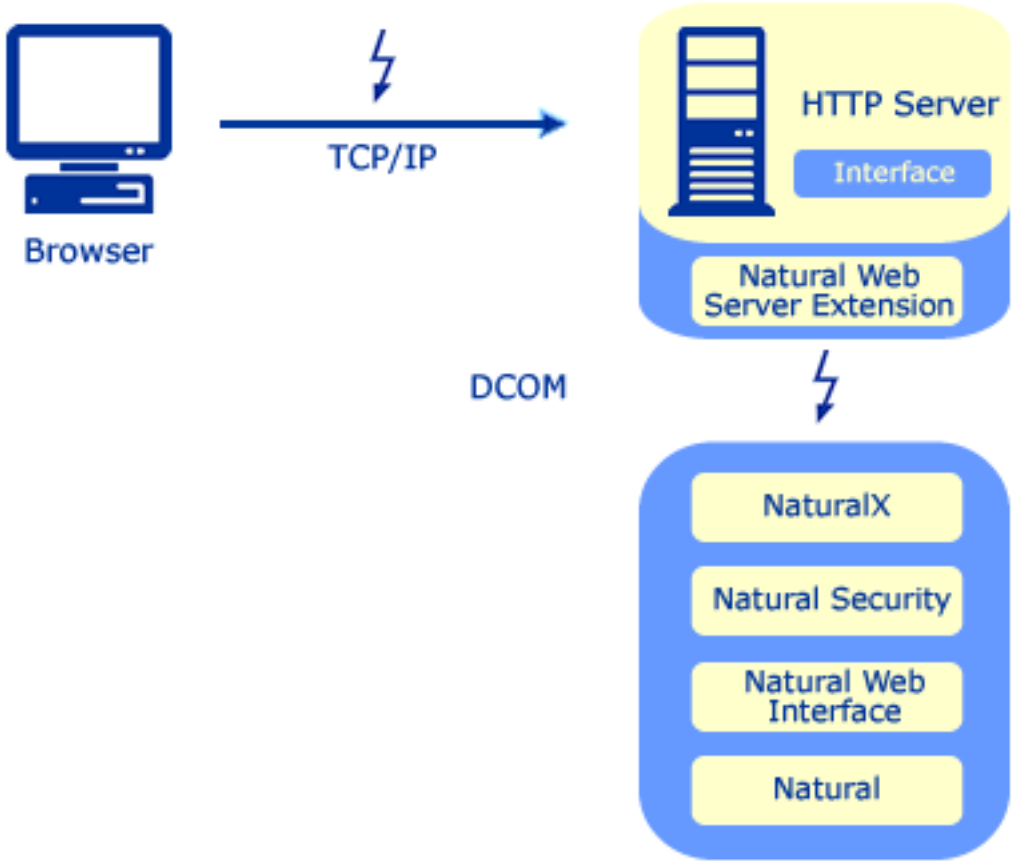
- Communication Using Natural RPC Techniques
- Communication Using DCOM Techniques

- Communication Using PAL Techniques

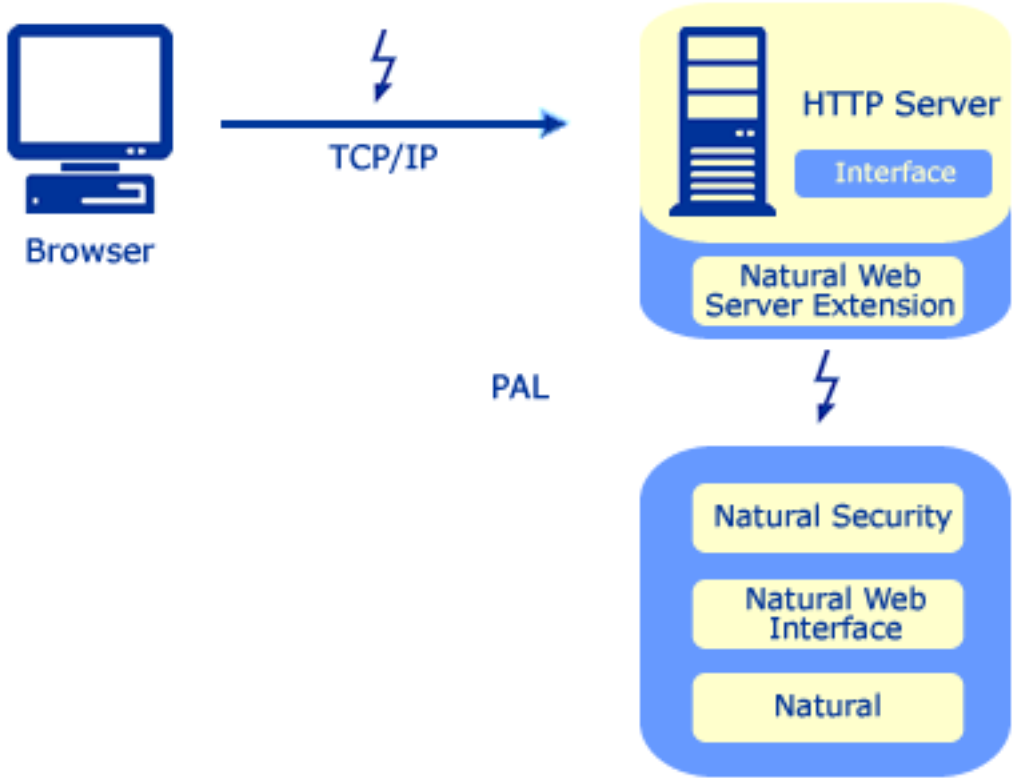
Communication Using Natural RPC Techniques



Communication Using DCOM Techniques



Communication Using PAL 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, PAL or DCOM technology

Natural Web Interface 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 - for example using forms on a web page that contain input fields and buttons. Users can enter and submit data via the forms which then executes a Natural subprogram that 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, 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: 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).

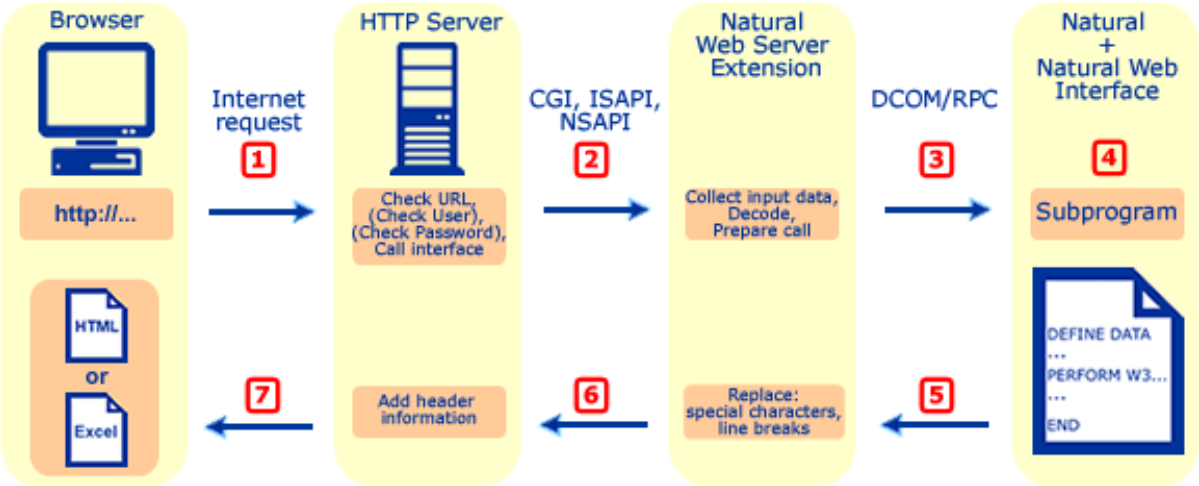
Web Page Creation

Web pages can be created with standard tools (e.g. Adobe Dreamweaver or Microsoft Expression Web) 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 of the Natural Web Interface

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 subrograms 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.

Natural Web Interface 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 Installation and Configuration

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

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

- Supported HTTP Servers 22
- Configuring RPC and RPC Server 22
- Configuring the DCOM Server 24
- Configuring the Web Interface 24
- Configuring the Web Interface for Apache 2.4.x (64bit) 28
- Configuring an HTTP Server 29
- Communication with Natural Security 29

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 at ***https://documentation.softwareag.com/***.

Supported HTTP Servers

Operating System	HTTP Server
Windows (Intel)	■ Microsoft Internet Information Server Version 6.0/7.5 ■ Apache Version 2.0.x ■ Apache Version 2.2.x ■ Apache Version 2.4.x (64bit)
(*)	■ Apache Version 2.0.x ■ Apache Version 2.2.x ■ Apache Version 2.4.x (64bit)

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:

- General RPC Configuration Setting for All Platforms using SYSWEB3
- Current Version of Natural for z/OS, Linux, or Windows

- EntireX Communicator / EntireX Developer's Kit

General RPC Configuration Setting for All Platforms using SYSWEB3

ACIVERS Settings

You are recommended to set the profile parameter ACIVERS to a value of 6 or above. Refer to Set the API Version in the Natural RPC documentation for further details.

MAXBUFF and MAX-MESSAGE-LENGTH Settings

You are recommended to set the profile parameter MAXBUFF on the Natural RPC server to a value of 1024 or above. This also requires setting the value of MAX-MESSAGE-LENGTH to 1048576 or above on the EntireX Broker that is included in EntireX Communicator.

Current Version of Natural for z/OS, Linux, or Windows

On Windows and Linux 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 z/OS 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 Linux (All Platforms)

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

Configuring the DCOM Server

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

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

This section covers the following topics:

- NaturalX Server

NaturalX Server

For all servers supporting the Natural Web Interface, add the libraries SYSWEB3 (or SYSWEB) and SYSEXT as steplibs, as described above in the section **Current Version of Natural for z/OS, Linux, or Windows**.

Configuring the Web Interface

The following topics are covered below:

- Natural Web Interface
- Natural Web Server Extensions for RPC
- Natural Web Server Extensions for DCOM
- Natural Web Server Extensions for MOD

Natural Web Interface

For z/OS, Windows, and Linux 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

Natural Web Server Extensions for MOD

- Using an RPC Server
- Using a DCOM Server
- Using a SPoD Server

Using an RPC Server

1. Install the Natural Web Server Extensions
2. Open the `..\conf\httpd.conf` file of the HTTP Server and add the the following new lines for the RPC Interface:

For Apache 2.0.x

```
...
LoadModule nww3_module modules/nww3mod2.dll
<Location /nww3/mod>
  AllowOverride None
  NWWiniFile "<yourRoot>/nww3/nww3mod.ini"
  SetHandler nww3
</Location>
...
```

For Apache 2.2.x

```
...
LoadModule nww3_module modules/nww3mod22.dll
<Location /nww3/mod>
  AllowOverride None
  NWWiniFile "<yourRoot>/nww3/nww3mod.ini"
  SetHandler nww3
</Location>
...
```

3. Specify additional files at the `/nww3` directory, if not only one service or broker are to be used.
4. If a static read of the .ini file is wanted (this influences the performance), add the line shown in *italics* to your `obj.conf`.

Using a DCOM Server

1. Install the Natural Web Server Extensions.
2. Open the `..\conf\httpd.conf` file of the HTTP Server and add the the following new lines for the DCOM Interface:

For Apache 2.0.x

```
...
LoadModule nww3d_module modules/nww3dmod2.dll
<Location /nww3d/mod>
  AllowOverride None
  NWWiniFile "<yourRoot>/nww3d/nww3dmod.ini"
  SetHandler nww3d
</Location>
...
```

For Apache 2.2.x

```
...
LoadModule nww3d_module modules/nww3dmod22.dll
<Location /nww3d/mod>
  AllowOverride None
  NWWiniFile "<yourRoot>/nww3d/nww3dmod.ini"
  SetHandler nww3d
</Location>
...
```

3. Specify additional files at the `/nww3d` directory, if not only one service or broker are to be used.
4. If a static read of the .ini file is wanted (this influences the performance), add the line shown in *italics* to your `obj.conf`.

Using a SPoD Server

1. Install the Natural Web Server Extensions.
2. Open the `..\conf\httpd.conf` file of the HTTP Server and add the the following new lines for the SPoD (PAL) Interface:

   For Apache 2.0.x

```
...
LoadModule nww3p_module modules/nww3pmod2.dll
<Location /nww3p/mod>
  AllowOverride None
  NWWiniFile "<yourRoot>/nww3p/nww3pmod.ini"
  SetHandler nww3p
</Location>
...
```

   For Apache 2.2.x

```
...
LoadModule nww3p_module modules/nww3pmod22.dll
<Location /nww3p/mod>
  AllowOverride None
  NWWiniFile "<yourRoot>/nww3p/nww3pmod.ini"
  SetHandler nww3p
</Location>
...
```

3. Specify additional files at the `/nww3p` directory, if not only one service or broker are to be used.

4. If a static read of the .ini file is wanted (this influences the performance), add the line shown in *italics* to your `obj.conf`.

Configuring the Web Interface for Apache 2.4.x (64bit)

This functionality is currently only applicable in conjunction with RPC (see also ***Configuring RPC and RPC Server***).

In order to use the SYSWEB3 functionality, the MOD approach or the CGI approach can be chosen. MOD means that the module provided by the product is linked to the Apache Web Server libraries in order to speed up performance. The following sections show what modifications are necessary in the configuration file *httpd.conf* to allow the MOD and the CGI approach.

- Using MOD
- Using CGI

Using MOD

1. Add the module `nww3mod24` to the module list:

```
LoadModule nww3_module modules/nww3mod24.dll
```

2. Add the location section, where also the path to the required *.ini* file is specified.

   For example:

```
<Location /nww3mod>
  AllowOverride None
  NWWiniFile "<yourPath>/nww3mod24.ini"
  SetHandler nww3
</Location>
```

3. Copy the file *nww3mod24.ini* to `<yourPath>`.
4. Copy the module *nww3mod24.dll* to the module folder of the Apache installation.

   This module enables RPC access to the SYSWEB3 libraries.
5. In the browser the following link is then valid:

   http://`<yourApacheServer>`/nww3mod/sysweb3/nat-info

Using CGI

1. Add the script alias. For example:

```
ScriptAlias /nww3cgi "cgi-bin/nww3cgi64.exe"
```

2. Copy the CGI module *nww3cgi64.exe* into the *cgi-bin* folder of the Apache Installation.

   This module enables RPC access to the SYSWEB3 libraries.

3. In the browser the following link is then valid:

   http://<*yourApacheServer*>/nww3cgi/sysweb3/nat-info

Configuring an HTTP Server

Windows (Internet Information Server 6.0 and 7.5)

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.	Web Interface 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 librarySYSWEB3.
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 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.
Test Utility WEB-ONL3 with SYSWEB3	Describes how to use the Test Utility with SYSWEB3.
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 SYSWEB3 contains all modules of the Natural Web Interface.

8 Working with the Natural Web Interface

- Setting up your Environment 36
- Building Subprograms in Natural 37

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 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 **SYSWEB3** the following prerequisites must be met:

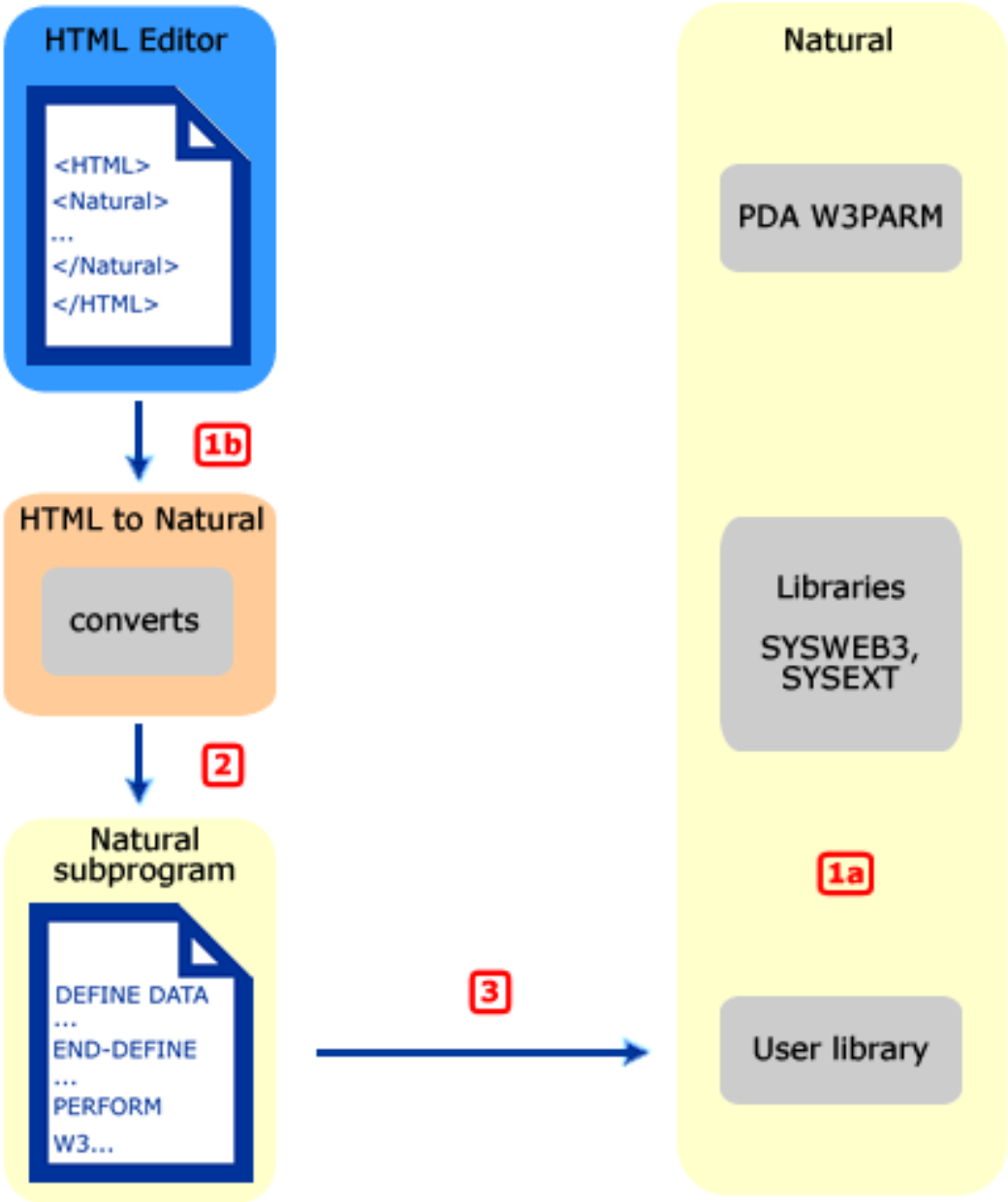
- Current Natural Version must be installed.
- The library SYSWEB3.
  Either Natural steplibs must be available or the contents of the library SYSWEB3 must be copied to the library SYSTEM or to the user library that will be called by the RPC.
- The parameter data Area W3PARM.
- The Natural RPC stub or 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 Web Interface Plug-in out of the library SYSWEB3 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 SYSWEB3.
- 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 SYSWEB3 subroutines (such as W3HTML or W3TEXT) for your return page in the program editor. See the programs in the library SYSWEB3, 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 librarySYSWEB3; the programs in the library SYSWEB3 enable you to perform basic system functions and in addition, the programs in the library SYSWEB3 generate HTML tags; this approach requires less explicit HTML knowledge and you can still modify the programs you are calling.

Example: Entering Calls to SYSWEB3 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 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 SYSWEB3 to evaluate how many bytes are free in the return page.

Testing Subprograms

There are three ways to test your subprograms:

1. Call the subprogram from your web browser.
2. Call the subprogram NAT-DIR in the library SYSWEB3 to see the contents of a Natural library. You can also specify the name of the library in the parameters, for example *http://.../sysweb3/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-ONL3 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 SYSWEB3

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.

You can use the `NAT-DOCU` subprogram to access the ***Natural Web Online Documentation SYSWEB3***.

Text Objects T3*

The `T3*` text objects describe the contents of the library SYSWEB3, what the subroutine names are and which parameters can be passed. They also provide a code sample of how to invoke them.

Subprograms E3*

Sample code of the online documentation.

Objects D3* and D4*

The `D3*` and `D4*` objects are demonstration applications.

Programs Web*

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

III Natural Web Server Extensions

This document comprises the following sections:

Introduction	Get an insight into the working and installation procedures of the Natural Web Server Extensions when using SYSWEB3.
Initialization File	Parameters and variables of the initialization file.
Error Messages	List of possible errors.

9 Natural Web Server Extensions - Introduction

- About Natural Web Server Extensions 56
- Installation - RPC / DCOM 56
- Transformations 57
- Variables 57
- Error Logging and Messages 57
- Calling Programs 58

This section covers the following topics:

About Natural Web Server Extensions

The Natural Web Server Extensions part is basically a program called from an HTTP server. The Natural Web Server Extensions program 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.

Three HTTP Server interfaces are supported:

- Common Gateway Interface (CGI), for supported server and platforms,
- Internet Server Application Programming Interface (ISAPI) only for Microsoft Internet Information Server on Windows.
- Apache Module (mod) only for Apache Server.

Installation - RPC / DCOM

Each Natural Web Server Extension consists of two files:

- an executable and
- an **initialization file**.

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

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

	RPC	DCOM
CGI	nww3cgi.exe nww3cgi.ini	nww3dcgi.exe nww3dcgi.ini
ISAPI	nww3isapi.dll nww3isapi.ini	nww3disapi.dll nww3disapi.ini
Apache 2.0.x (Linux)	nww3mod2.so nww3/mod	nww3dmod2.so nww3d/mod
Apache 2.0.x (PC)	nww3mod2.dll nww3/mod	nww3dmod2.dll nww3d/mod
Parameters	`RPC_ETB_ID_NAME` = broker name `RPC_SERVER_NAME` = service name	`NWW_INOUT_LENGTH` = amount of transferred data

	RPC	DCOM
	NWW_INOUT_LENGTH = amount of transferred data	

Note: On some HTTP servers executable files without the extension .exe can be used.

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 required.

Variables

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

Each variable to be transferred requires 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 the files of the Natural Web Server Extension are located. 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 (PC)	`http://server-name/cgi-library/nww3cgi.exe/your-library/your-program`	`http://server-name/c`
ISAPI	`http://server-name/cgi-library/nww3isapi.dll/your-library/your-program`	`http://server-name/c`
mod	`http://server-name/nww3/mod/your-library/your-program`	`http://server-name/n`

10 Natural Web Server Extensions - Initialization File

- About the Initialization File 60
- RPC Parameters 60
- PAL Parameters 61
- DCOM Parameters 61
- Natural Web Server Extension Settings 61
- Data Transfer Settings 63
- HTTP Server Variables 64
- Additional Variables 65
- Error Templates 65

This section covers the following topics:

About the Initialization File

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.

PAL Parameters

These parameters are required for communication with the SPoD (PAL) interface.

Parameter	Description
PAL_SERVER_NODE	Name of the called PAL server node.
PAL_SERVER_PORT	Number of the called PAL server port.
PAL_SESSION_PARAMETER	If dynamic parameters are required for your server, specify the session parameters using single quotes ('...').

DCOM Parameters

This parameter is required for the communication with DCOM.

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

Natural Web Server Extension Settings

This group of parameters 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 2013] nwwcgi.exe 04.02.13 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.

Parameter	Description
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	Default values for the SYSWEB3 interface are: ■ NWW_INOUT_FORMAT=0 ■ NWW_INOUT_LENGTH=0
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:

Parameter	Description

	(Default value for ASCII)
	
	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.

Data Transfer Settings

This group of parameters defines the data transfer to the Natural server.

Parameter	Description
SETMIMETYPE	This parameter defines the transfer type for an incoming mime-type. The definition contains the name of a mime-type and the appropriate transfer type. The mime-type can contain asterisks as wildcard for one or any character. The application always uses the first matching mime-type. Valid transfer types are binary or alpha. Examples: SETMIMETYPE=text/*;*charset*=*utf-16 binary SETMIMETYPE=text/* alpha SETMIMETYPE=image/svg alpha SETMIMETYPE=image/* binary SETMIMETYPE=multipart/form-data* binary SETMIMETYPE=* alpha **Note:** The line SETMIMETYPE=* alpha defines the default setting, for all unknown mime-types, if not set, alpha is used.

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 see **https://docs.microsoft.com/en-us/iis/web-dev-reference/server-variables**.

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.softwareag.com: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.
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>
```

11 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 inst
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 installati
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 installati
NWW0012	ERX error register.	EntireX RPC Service cannot be initialized.	None.	None.	Check configura

Error Number	Error Message	Description	User	Programmer	Administrator
NWW0013	erx.dll cannot be loaded. Subcode:	EntireX erx.dll not found.	None.	None.	Check installation.
NWW0014	ERX logon failed.	EntireX logon cannot be performed.	Check User-ID, Password	Check installation file for Check User-ID, Password.	Check installation.
NWW0015	ERX logoff failed.	Logoff from EntireX failed.	None.	None.	Contact support.
NWW0033	File ... not found (Error: ...).	The initialization file for your adapter cannot be found.	None.	None.	Check your obj.conf setu for NSAPI.
NWW0034	NWW_USER_ID too long.	User ID only with a maximum of 8 characters allowed.	None.	None.	Specify only user IDs wi 8 characters or fewer, ev if other system will allo more.
NWW0035	NWW_PASSWORD too long.	Passwords only with a maximum of 8 characters allowed.	None.	None.	Specify only (user-) passwords with 8 charact or fewer, even if other system will allow more.
NWW0036	Natural Library Name too long.	Natural allows only library names up to 8 characters.	Check URL.	Check URL specification.	None.
NWW0037	Natural Subprogram Name too long.	Natural allows only Subprogram names up to 8 characters.	Check URL.	Check URL specification.	None.
NWW0099	CONTENT_TYPE: ... is not supported.	Only data with CONTENT_TYPE = application/x-www-form-urlencoded is supported.	None.	Do not use the attribute ENCTYPE at your FORM tag.	None.
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.
NWW0815	Interface A1(1:v) no longer supported.	Wrong interface specified.	None.	None.	Remove NWW_INOUT_FORMA and NWW_INOUT_LENGT 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.

Error Number	Error Message	Description	User	Programmer	Administrator
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 support
NWW1006	Unicode ASCII conversion failed.	The transferred data has to be converted.	None.	None.	Contact support
NWW1007	Method ... not found.	A specified method cannot be called.	Correct the URL.	Add method to your class.	Check your regi configuration.
NWW1008	Class ... not found.	A specified class cannot be called.	Correct the URL.	Create class and register with REGISTER *	Check your regi configuration.
NWW1009	Initialization of Class ... failed.	A specified class cannot be called.	Correct the URL.	Create class and register with REGISTER *	Check your regi configuration.
NWW1010	DCOM call failed, error	The call to DCOM failed.	None.	None.	Check your con
NWW1011	DCOM error ... occurred ...	Internal DCOM error. See DCOM documentation for further information.	Correct the URL.	Correct the program.	Correct the Congiguration/I
NWW1012	DCOM initialization failed.	The initial call to DCOM failed.	None.	None.	Correct the Configuration/I
NWW1013	DCOM release failed.	The deletion of class and close of DCOM failed.	None.	None.	Correct the Configuration/I
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.
NWW2009	Logon to Library ... failed (Error: ...).	The specified library name is not correct.	Correct the URL.	None.	None.
NWW2010	PAL call failed (Error: ...).	The call to PAL failed.	None.	None.	Check your con
NWW2011	PAL error ... occurred.	Internal PAL error. See PAL documentation for further information.	Correct the URL.	Correct the program.	Correct the Congiguration/I
NWW2012	PAL initialization failed (Subcode: ...).	The initial call to PAL failed.	None.	None.	Correct the Configuration/I
NWW2013	PAL Transport not initialized. (Subcode: ...).	Natural allows only subprogram names up to 32 characters.	Check the URL.	Check the URL specification.	None.

Error Number	Error Message	Description	User	Programmer	Administrator
NWW2014	PAL Transport failed. (Subcode: ...).	Natural allows only subprogram names up to 32 characters.	Check the URL.	Check the URL specification.	None.

12 Test Utility WEB-ONL3 with SYSWEB3

- Prerequisites 74
- Running the Application 74
- Supported Content Types 75
- Input/Output Fields 76

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

Features:

- Local application checking.
- No need for remote debugging.
- Simplified error checking.
- Comfortable operation by user friendly interface.
- No need to write an extra test routine.

This section covers the following topics:

Prerequisites

- Web browser which supports different content types, for example, Microsoft Internet Explorer Version 5.0 or higher.
- Any available text editor.

Running the Application

To define path adjustments

1. Start the main dialog.
2. Select a browser and viewer of your choice with Tools > Options...
3. Set the browser, viewer and work file path.
4. Press the OK button.

To start the application

1. Start the dialog **WEB-ONL3**.
2. Select a library and subprogram name.

3 Optional: add parameters.

4 Choose RPC or DCOM.

5 Press the Execute button.

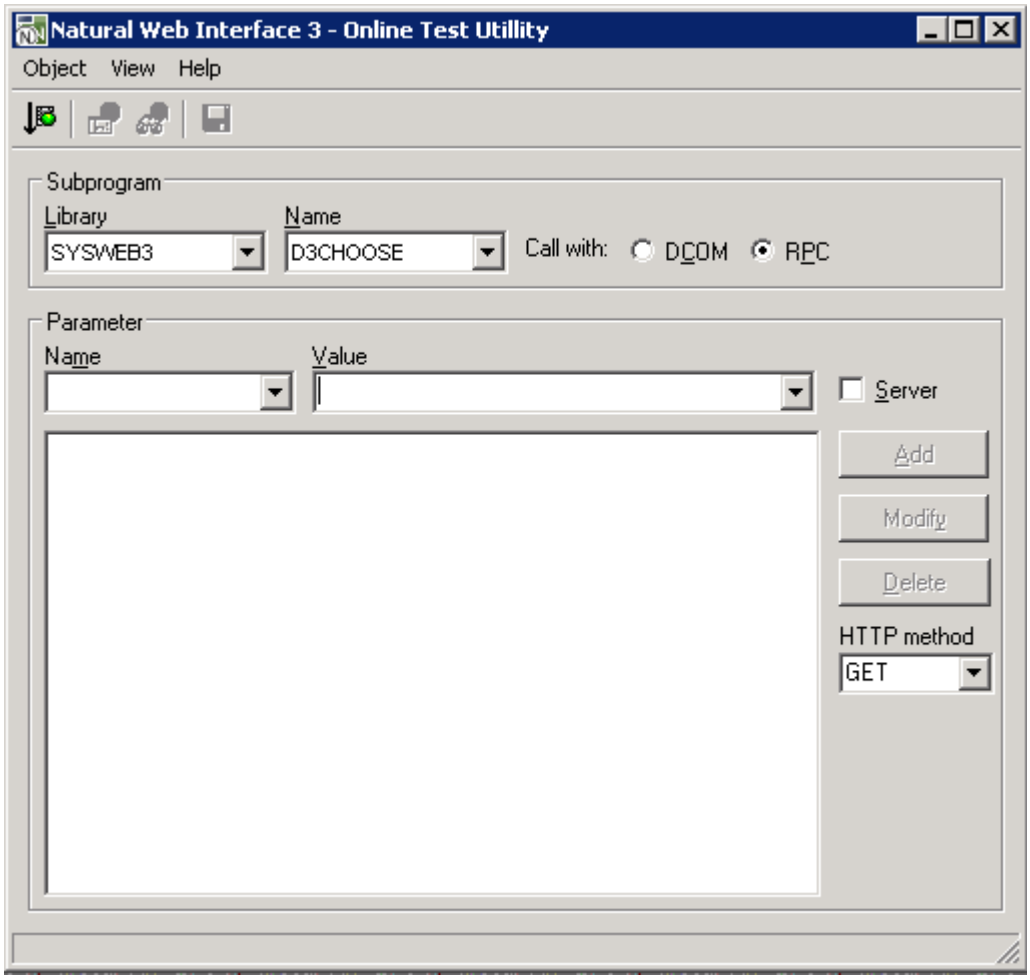
6 View the result by pressing either the Result... or the Browse... button.

Supported Content Types

The following Content Types are supported by the Test Utility:

Content Type	Extension
"application/rtf"	"rtf"
"application/powerpoint"	"ppt"
"application/msword"	"doc"
"application/excel"	"xls"
"text/html"	"htm"
"text/plain"	"txt"
"text/xml"	"xml"
"text/richtext"	"rtf"

If you need further Content Types, change the subroutine HTML2CONTENT-TYPE (SYSWEB3/W3CO2EXT) and extend the translation table to your own needs.



Input/Output Fields

Field	Explanation
Subprogram: Library Name	Enables you to specify the library and the name of the required subprogram. The available libraries and subprograms are automatically taken from the library workspace and listed in selection boxes.
Interface	Can be selected with either DCOM or RPC as communication form. For DCOM, you have to register your classes first. Default: RPC
Parameters: Name Value Server	Here you can enter the name-value-pairs required from the subprogram. To add them to the parameter list, use the Add button. To modify the entries, use the Modify button. You do not have to substitute &, =, %; this will be done by the WEB-ONL3 program. If you use server parameters, check the Server toggle button before you add the parameter to the parameter list.

Field	Explanation
	In the parameter list, all name-value-pairs are displayed. &, =, % are substituted. To delete a pair, select the item and press the Delete button. Every selected item will be inserted into the Name and Value fields. If you wish to modify a pair, select the item, change it in the Name and Value fields and press the Modify button. Server: If any of the name-value-pairs are server variables, you need to check this toggle button. Note that any status will last until you change it again.
HTTP Method	In this drop-down list you can select the HTTP request/submit method to be used: ■ **HEAD** Identical to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content. ■ **GET** Requests a representation of the specified resource. ■ **POST** Submits data from the identified resource. The data is included in the body of the request. You can use this method to submit data with a different content type, for example XML files or binary data (such as graphics). If you specify this method, an additional **Browse...** button and the **Binary** checkbox are available on the screen. Use the **Browse...** button to choose a file and the **Binary** checkbox, if you want to submit binary data. If you specify this method without an input file, the default mime type "application/x-www-form-urlencoded" is set. If you use an input file, the content type of that file will be used, for example with an XML file, the content type will automatically be set to "text/xml". You can specify a different mime type in the input field manually. **Note:** A mime type which has been set manually will always override the default mime type. ■ **PUT** Uploads a representation of the specified resource. You can use this method to submit data with a different content type, for example XML files or binary data (such as graphics). If you specify this method, an additional **Browse...** button and the **Binary** checkbox are available on the screen. Use the **Browse...** button to choose a file and the **Binary** checkbox, if you want to submit binary data.

Object

Execute Subprogram
Starts the editor chosen with the Options dialog.

Save to Natural Text
Saves the returned data as Natural object of the type Text.

Exit
Leaves the dialog.

View

Result...
Executes the selected subprogram.

Browser...
Starts the browser chosen with the Options dialog.

Options...
Opens the Options dialog.

Help

Contents
Displays this HTML-based help file.

About
Provides general program information.

13 Programming Tips

- Editing in Lower Case 80
- Quote vs. Apostrophe 80
- Variables defined by Value 81
- Access to Resources 81
- Constant Values 81
- Creating a New Page 82
- DCOM / RPC 82

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 z/OS, 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 setting of keyword subparameter TQMARK of profile parameter CMPO or macro NTCMPO. This subparameter controls the translation of a quotation mark (") within a Natural text constant. It takes effect at compilation time only. Set this subparameter 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.
- 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.
- To generate a class for a Natural library, use the **Class Generation** in the Web Interface Plug-In.

14 Web Interface Administration

- Create a User-Defined Error Page 84
- Create a User-Defined Error Page XML-Style 84
- Alphanumeric-to-HTML Conversion 84
- Alphanumeric-to-URL Conversion 85

This section covers the following topics:

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 (SYSWEB3/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 (SYSWEB3/W3ERRTMP).
2. Open the subroutine SYSWEB3/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 (SYSWEB3/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

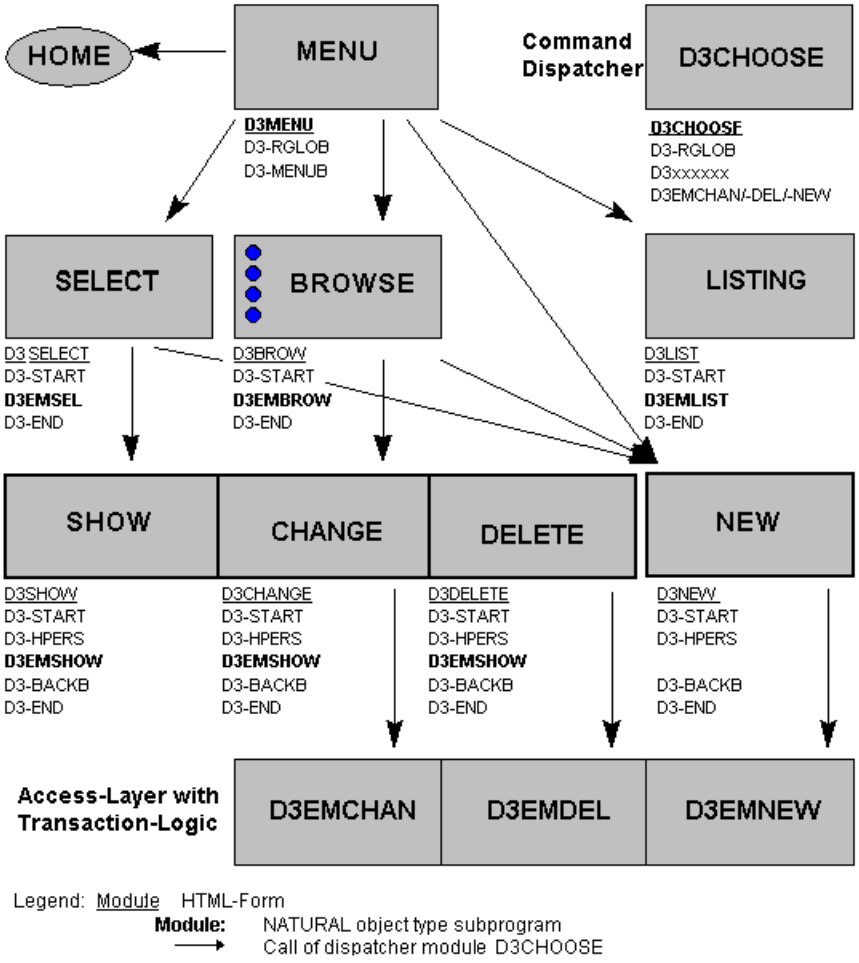
- About the Demo Application without JavaScript 88
- Design Decisions 89
- Libraries, Modules, and Naming Conventions 89
- Starting the Demonstration Application D3MENU 90
- Starting the Natural Web Interface Online Manual 90
- Requirements for the D3MENU Demonstration Application 90

This section covers the following topics:

About the Demo Application without JavaScript

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**):

SYSWEB3

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 D3MENU

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 SYSWEB3.

Example of the URL to call the demonstration application:
`http://yourserver/yourcgi/sysweb3/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 SYSWEB3.

Example of the URL to call the demonstration application:
`http://yourserver/yourcgi/sysweb3/d3menu`

Requirements for the D3MENU Demonstration Application

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 SYSWEB3 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

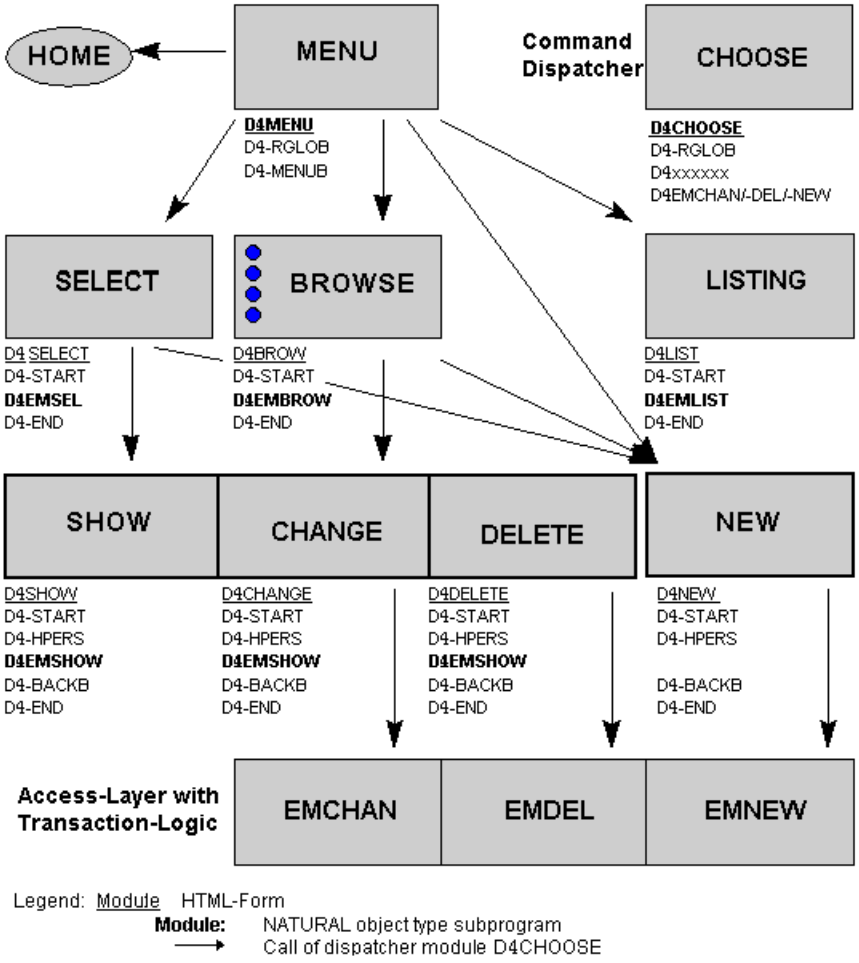
- About the Demo Application with JavaScript 92
- Design Decisions 93
- Starting the Demonstration Application D4ENTER 93
- Requirements 93

This section covers the following topics:

About the Demo Application with JavaScript

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 D4ENTER

The start module for the demonstration is D4ENTER. Depending on your installation of the Natural Web Server Extension, call the subprogram D4ENTER in library SYSWEB3.

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 SYSWEB3 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

- Error Messages 96

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 Migrate Natural Web Interface SYSWEB to SYSWEB3

This section provides you with a step-by-step introduction on how to update your programs written for the Natural Web Interface (SYSWEB) to run with the new interface (SYSWEB3).

1. You have to use the new interface programs `nww3*` instead of the old `nww*` programs. Copy the appropriate programs to your HTTP server, and configure your HTTP server.
2. Adapt your nww3*.ini file settings.
3. Adapt external URLs to `nww3*` instead of `nww*`

   Note: It is possible to rename the `nww3*` interface programs - but the new interface programs will only work with Natural subprograms stowed with SYSWEB3 and the old interface programs will only work with Natural subprograms stowed with SYSWEB.

4. If you are running a Natural RPC server
   - On Windows or Linux without Natural Security, modify your steplib settings at NATPARM. Replace steplib SYSWEB with steplib SYSWEB3 (or use a new, different RPC server instead)
   - On z/OS systems without Natural Security, modify your server startup logon from SYSWEB to SYSWEB3 (or use a new, different RPC server instead)
   - With Natural Security (all platforms), modify your steplib settings at Natural Security. Replace steplib SYSWEB with steplib SYSWEB3 (or use a new, different RPC server instead)
   - Check your EntireX Broker settings. You may have to expand your `NUM-COMBUF`, `NUM-LONG-BUFFER`, `MAX-MSG` settings, depending on the maximum page length you want to transfer.
5. If you are running a DCOM server (Windows only),
   - Unregister your class (usually named according to your library name) first.
   - Delete your class (usually named according to your library name) and the associated LDA (usually named L)

- Use the Web Interface plug-in and generate a new class for SYSWEB3 with a new LDA (new GUIs).
- Register the new class.

6. Adapt URLs, read work files, templates, etc. to point to SYSWEB3 instead of SYSWEB.

   Note: If you rename SYSWEB3 to SYSWEB your application may work, but internal programs delivered with SYSWEB3 may not work correctly and may deliver wrong or unexpected results.

7. Recatalog all your programs that reference web interface programs by using SYSWEB3 as steplib instead of SYSWEB.
8. All your programs should now run the same way, they did with SYSWEB before.

IV Web Interface Plug-In

19 Web Interface Plug-In

- Before You Start 102
- Invoking Web Interface Plug-In Commands 102
- Web Interface Plug-In Functions 103

This section covers the following topics:

Before You Start

The Web Interface Plug-In is an optional plug-in unit for Natural Studio. Therefore, before any action can be taken, the Web Interface Plug-In must be activated in your Plug-In Manager. For detailed information on the activation procedure, see the section Plug-In Manager.

Web Interface Plug-In Interface

Once the Web Interface Plug-In is activated, your Natural Studio interface will be changed in the following way:

- the Tools menu will display the relevant Web Interface Plug-In commands,
- a toolbar will be available for the commands most frequently used.

The commands available depend on your working context.

Invoking Web Interface Plug-In Commands

To invoke Program Generation commands from the main menu

- From the main menu, choose **Tools > Development Tools > Web Interface Program Generation...**

The available commands are described in detail in the section **Program Generation**.

To invoke Class Generation commands from the main menu

- From the main menu, choose **Tools > Development Tools > Web Interface Class Generation...**

The available commands are described in detail in the section **Class Generation**.

To invoke Test Utility commands from the main menu

- From the main menu, choose **Tools > Development Tools > Web Interface Test Utility...**

The available commands are described in detail in the section **Online Test Utility**.

To invoke Web Interface options from the main menu

- From the main menu, choose **Tools > Development Tools > Web Interface Options...**

The available commands are described in detail in the section **Options**.

To invoke Web Interface Plug-In commands, use the following toolbar buttons

- For the Generation Wizards:

  Selects the HTML page that should be used for the generation process.

  Starts the external editor for the selected HTML file.

  Or:

  For the Test Utility:

  Starts the editor.

  It is disabled as long as you have not executed the program and if you have not changed the subprogram library or name.

  You can choose the editor within the **Options** dialog.

  Starts the browser..

  It is disabled as long as you have not executed the program and if you have not changed the subprogram library or name.

  You can choose the browser in the **Options** dialog.

Web Interface Plug-In Functions

Program Generation

Note: The Program Generation Wizard is not applicable to z/OS systems.

This section describes the use of the Program Generation Wizard, a plug-in that enables you to generate basic web Interface programs and programs that use HTML templates with the Natural Web Interface.

The basic generation can be used to generate necessary parts for a subprogram called from the internet with the web interface. Then your specific coding can be added.

The template generation works with ready designed HTML pages. These HTML pages will be loaded from the resource directory. Then specific parts can be replaced with your individual parts. The program generator reads these HTML pages, searches for the parts to be replaced (marked with special characters) and then generates an external subroutine that can be used equal to output only maps.

This section covers the following topics:

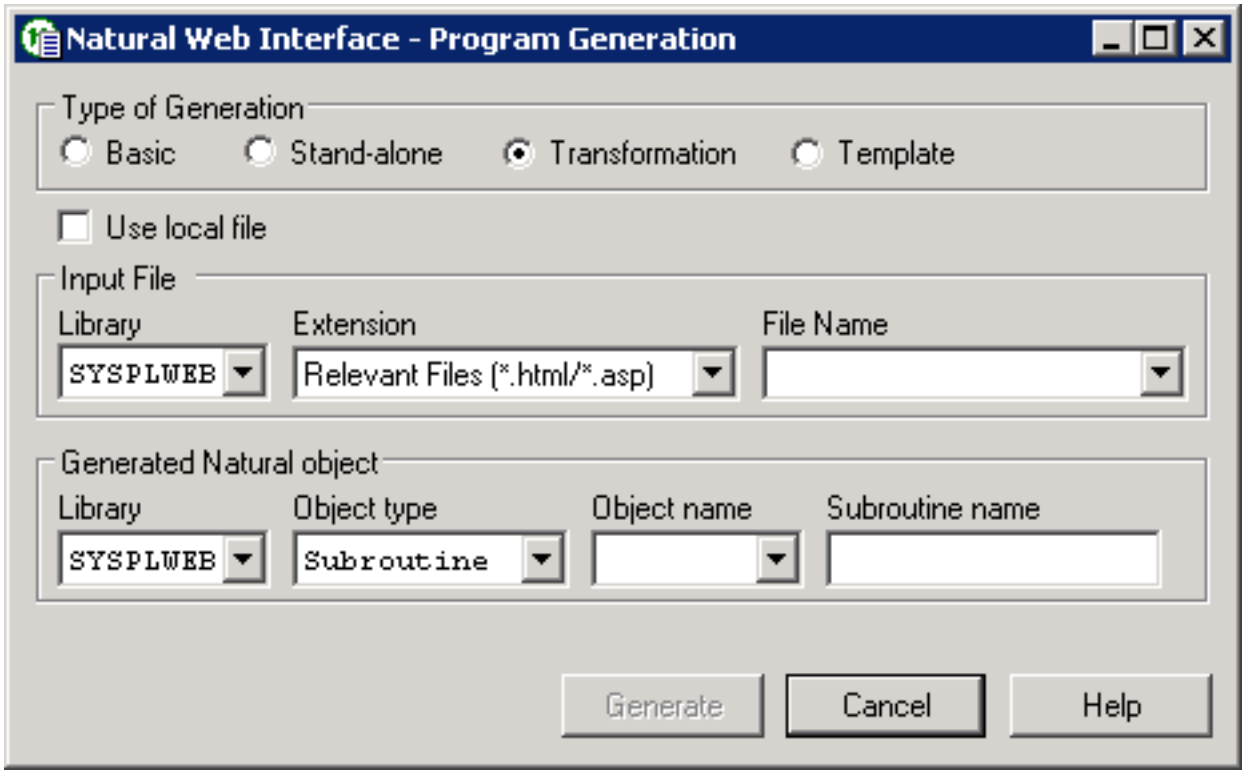
- Using the Conversion Program
- Inserting Replacement Strings
- Options
- View
- Class Generation
- Online Test Utility

Using the Conversion Program

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

The Web Wizard is a dialog that uses an HTML page as input and generates a Natural subprogram, which can be called by the Natural Web Server Extensions using the Natural Web Interface, or a subroutine which can be called to generate the output.

With the basic generation of the Web Wizard, a standalone Natural subprogram that can be called by the Natural Web Server Extensions using the Natural Web Interface can be produced.



Generating a Basic Subprogram to be Called Directly from the Web

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

1. Select Type of generation: Basic.
2. Select your Generated Natural object.
3. Start the generation.
4. If you generated this subprogram the first time and you want to call the generated subprogram via DCOM, regenerate the DCOM class (see: **Class Generation**).
5. After the generation, this page can be called from the internet, but because this page does not set any data, the page will be empty.

Example of a basic generation

Generated Natural subprogram, to be called directly from the internet:

```
0010 * ----- GENERATED BY NATURAL WEB INTERFACE
* Library .....: SYSPLWEB
* Source Name .: BASIC
* -----
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
* ----- PRIVATE VARIABLES -----
* LOCAL
* 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
* --- READ ENVIRONMENT ---
* PERFORM W3READ-ENVIRONMENT-DYNAMIC 'varname' ' ' W3VALUE
* set default value
* IF *length(W3VALUE) = 0 THEN
*  W3VALUE := ??
* END-IF
* ----- HEADER FOR SERVER -----
* PERFORM W3CONTENT-TYPE 'text/html'
*
*
* Add your individual coding using W3* subroutines or
* call your own subroutines.
*
*
* ----- END HTTP -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
END
```

Generating a Standalone Subprogram to be Called Directly from the Web

To generate a subprogram to be called directly from the Web:

1. Select Type of generation: Standalone.
2. Select your Generated Natural object.
3. Start the generation.
4. If you generated this subprogram for the first time and you want to call the generated subprogram via DCOM, regenerate the DCOM class (see: **Class Generation**).

5 After the generation, you can call the Natural Web Interface to show the page.

Example of a standalone generation

Generated Natural subprogram, to be called directly from the internet:

```
* ----- GENERATED BY NATURAL WEB INTERFACE
* Library .....: SYSPLWEB
* Source Name .: ALONE
* -----
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
* ----- PRIVATE VARIABLES -----
LOCAL
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'
*
* --- READ ENVIRONMENT ---
* PERFORM W3READ-ENVIRONMENT-DYNAMIC 'varname' ' ' W3VALUE
*  set default value
* IF *length(W3VALUE) = 0 THEN
*  W3VALUE := ??
* END-IF
* --- WRITE THE HEAD OF THE DOCUMENT ---
PERFORM W3TEXT "<!DOCTYPE 'HTML PUBLIC-//W3C//DTD HTML 3.2//EN'>"-
'<html>'-
'<head>'-
"<meta http-equiv='Content-Type' content='"-
"text/html; charset=iso-8859-1'>"-
'<title>SYSPLWEB/TEST</title>'-
'</head>'
* --- WRITE THE BODY OF THE DOCUMENT ---
PERFORM W3TEXT '<body>'-
'<h2>SYSPLWEB/TEST</h2>'-
'<hr>'
*
PERFORM W3TEXT '<p>This is your output</p>'
*
COMPRESS '<hr>generated:' *DATE *TIME INTO W3VALUE
```

```
PERFORM W3TEXT W3VALUE
* --- END THE BODY OF THE DOCUMENT ---
PERFORM W3TEXT '</body>'-
'</html>'
*
* ----- END HTTP -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
END
```

Generating a Subprogram/Subroutine using Natural Tags

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

1. Select Type of generation: Transformation.
2. Select your input file of type HTML.
3. You can view your selected HTML page with an editor/browser.
4. Choose the Natural library you want to generate to.
5. Select the object type you want to generate.
6. Select your Generated Natural object.
7. Start the generation.
8. If you generated this subprogram for the first time and you want to call the generated subprogram via DCOM, regenerate the DCOM class (see: **Class Generation**).
9. After the generation, you can call the Natural Web Interface to show the page.

Inserting a Natural Tag

It is possible to specify Natural coding directly in the HTML page. After generation, the program needs no additional changes.

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

Appearance

```
<NATURAL> </NATURAL>
```

Below is information on:

- Attributes DATA, LDA, GDA, SUB, NOT
- Comment Tag
- ASP-like Script Commands

- Additional Script Directives
- Example of a Simple Generation
- Example of a Simple Generation with a Natural Tag

Attributes DATA, LDA, GDA, SUB, NOT

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

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

Comment Tag

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

ASP-like Script Commands

Not only `<NATURAL>` and `</NATURAL>` can be used, but also ASP-like (Active Server Pages) script commands which are differentiated from the text by using the `<%` and `%>` delimiters.

Additional Script Directives

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

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

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

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

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

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

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

Example 1 of a Simple Generation

HTML document:

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

Generated Natural subprogram:

```
* ----- GENERATED BY NATURAL WEB INTERFACE
* File ..........:
    E:\SAG\Natural\v.r\Fnat\SYSWEB\RES\example1.html
* Library .......: SYSWEB
* Source Name ...: EXAMPLE1
* Crunch Lines...: 1
* Save Source....: 1
* Line Length....: 128
* Long Constants.: 1
* -----
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
LOCAL
* ----- PRIVATE VARIABLES -----
1 W3VALUE (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALIZE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
```

```
* ----- MAIN PROGRAM -----
PERFORM
      W3TEXTLINE'<HTML><HEAD><TITLE>'
PERFORM W3TEXTLINE 'Example genNat'
PERFORM
      W3TEXTLINE'</TITLE></HEAD><BODY><H2>'
PERFORM W3TEXTLINE 'Example genNat'
PERFORM W3TEXTLINE '</H2><HR>'
PERFORM W3TEXTLINE '<P>This is for your output'
PERFORM W3TEXTLINE '</BODY></HTML> '
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
*
* ----- SUBROUTINES -----
END
```

Example 2 of a Simple Generation with a Natural Tag

HTML document:

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

Generated Natural subprogram:

```
* ----- GENERATED BY NATURAL WEB INTERFACE
* File ..........: E:\SAG\Natural\v.r\Fnat\SYSWEB\RES\example2.html
* Library .......: SYSWEB
* Source Name ...: EXAMPLE2
* Crunch Lines...: 1
* Save Source....: 1
* Line Length....: 128
* Long Constants.: 1
* -----
DEFINE DATA
PARAMETER USING W3PARM
LOCAL USING W3CONST
1 #VALUE (A30)
* ----- PRIVATE VARIABLES -----
1 W3VALUE (A250)
END-DEFINE
*
* ----- ERROR HANDLER -----
ON ERROR
  PERFORM W3ERROR ##W3ERROR
  PERFORM W3END ##RPC
  ESCAPE ROUTINE
END-ERROR
* ----- INITIALIZE HTTP API -----
PERFORM W3INIT ##RPC
* ----- HEADER FOR SERVER -----
PERFORM W3CONTENT-TYPE 'text/html'
*
* ----- MAIN PROGRAM -----
PERFORM W3TEXTLINE'<HTML><HEAD><TITLE>'
PERFORM W3TEXTLINE 'Example2 genNat'
PERFORM W3TEXTLINE'</TITLE></HEAD><BODY><H2>'
PERFORM W3TEXTLINE 'Example2 genNat'
PERFORM W3TEXTLINE '</H2><HR>'
PERFORM W3TEXTLINE '<P>This is for your output'
PERFORM W3TEXTLINE '<HR>'
PERFORM W3TEXTLINE '<P>generated at:'
 PERFORM DOTIME
PERFORM W3TEXTLINE '</BODY></HTML>'
* ----- END HTTP API -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
*
* ----- SUBROUTINES -----
DEFINE SUBROUTINE DOTIME
 COMPRESS *TIME *DATE INTO #VALUE
 PERFORM W3TEXTLINE #VALUE
END-SUBROUTINE
END
```

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

Generating a Subprogram/Subroutine using a Template that is Called Directly from the Web

To generate a subprogram/subroutine using a template that is called directly from the Web:

1. Select type of generation: Template.
2. Select your input file of type HTML.
3. You can view your selected HTML page with an editor/browser.
4. Select the object type you want to generate.
5. Select your Generated Natural object.
6. Start the generation.
7. If you generated this subprogram the first time and you want to call the generated subprogram via DCOM, regenerate the DCOM class (see: **Class Generation**).
8. After generation, you can call the Natural Web Interface to show the page.

Inserting Replacement Strings

It is necessary to specify the replacement strings directly in the HTML page. The replacement strings have to start and end with an specific character, e.g. $ (see **Options**). The name (content) of a string has to comply with the Natural rules for variable names. If not, subroutines may not stow.

If the name of the replacement string is prefixed with "HTML", unsaved characters as "<" or ">" will be replaced during replacement at runtime.

The following prefixes for automatic conversion at runtime are implemented:

- HTML
- URL
- XML

For more information, see the documentation of the subroutine **W3REPLACE-AT-OUTPUT**.

Example of Template Generation

HTML document:

```
<HTML>
<HEAD>
  <TITLE>Template Processing</TITLE>
</HEAD>
<BODY>
<H2>
  Template Processing
</H2>
<P>
  <HR>
<TABLE BORDER="0">
<TR><TD>Log-Time:</TD><TD>$log$<TD></TR>
<TR><TD>HTTPs Extension:</TD><TD>$html-ext$<TD></TR>
<TR><TD>Web Interface:</TD><TD>$html-ver$<TD></TR>
</TABLE>
<P>
<TABLE BORDER='0' WIDTH='100%' CELLSPACING='0' CELLPADDING=5>
  <TR BGCOLOR='#00cc66'>
    <TD>$prog$ - $log$</TD>
    <TD ALIGN='RIGHT'>Natural</TD>
  </TR>
</TABLE>
</BODY></HTML>
```

Generated Natural subroutine, that has to be called from a subprogram that is called from the internet:

```
* ----- GENERATED BY NATURAL WEB INTERFACE
* File ........: E:\SAG\Natural\v.r\Fnat\SYSWEB\RES\templ.html
* Library .....: SYSWEB
* Source Name .: TEMPL
* Delimiter ...: $
* -----
DEFINE DATA PARAMETER
1 log                                (A) DYNAMIC BY VALUE
1 html-ext                           (A) DYNAMIC BY VALUE
1 html-ver                           (A) DYNAMIC BY VALUE
1 prog                               (A) DYNAMIC BY VALUE
END-DEFINE
*
*
DEFINE SUBROUTINE e3templm
*
* ----- HEADER FOR SERVER -----
PERFORM W3CLEAR
```

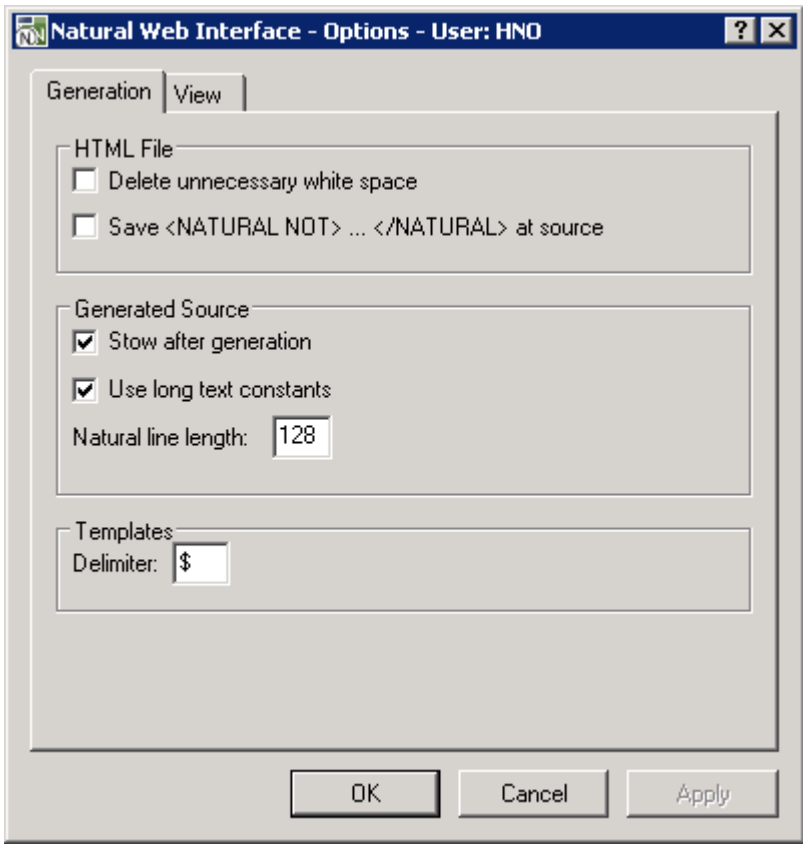
```
PERFORM W3CONTENT-TYPE 'text/html'
* ----- MAIN PROGRAM -----
* --- LOAD THE HTML TEMPLATE ---
PERFORM W3LOAD-RESOURCE ' '   'e3templ.html'
*
* --- REPLACE PLACEHOLDER ---
PERFORM W3REPLACE-AT-OUTPUT ' ' '$log$' log
PERFORM W3REPLACE-AT-OUTPUT 'HTML' '$ext$' ext
PERFORM W3REPLACE-AT-OUTPUT 'HTML' '$ver$' ver
PERFORM W3REPLACE-AT-OUTPUT ' ' '$prog$' prog
* ----- END MAIN PROGRAM -----
*
END-SUBROUTINE
*
END
```

Generated Natural subprogram, to be called directly from the internet:

```
* ----- GENERATED BY NATURAL WEB INTERFACE
* File ........: E:\SAG\Natural\v.r\Fnat\SYSWEB\RES\templ.html
* Library .....: SYSWEB
* Source Name .: TEMPL
* Delimiter ...: $
* -----
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'
*
* ----- MAIN PROGRAM -----
* --- LOAD THE HTML TEMPLATE ---
PERFORM W3LOAD-RESOURCE 'SYSWEB'   'e3templ.html'
*
* --- REPLACE PLACEHOLDER ---
PERFORM W3REPLACE-AT-OUTPUT ' ' '$log$' 'replace-string-1'
PERFORM W3REPLACE-AT-OUTPUT 'HTML' '$ext$' 'replace-string-2'
PERFORM W3REPLACE-AT-OUTPUT 'HTML' '$ver$' 'replace-string-3'
```

```
PERFORM W3REPLACE-AT-OUTPUT ' ' '$prog$' 'replace-string-4'
* ----- END HTTP -----
PERFORM W3END ##RPC
* ----- END MAIN PROGRAM -----
*
END
```

Options

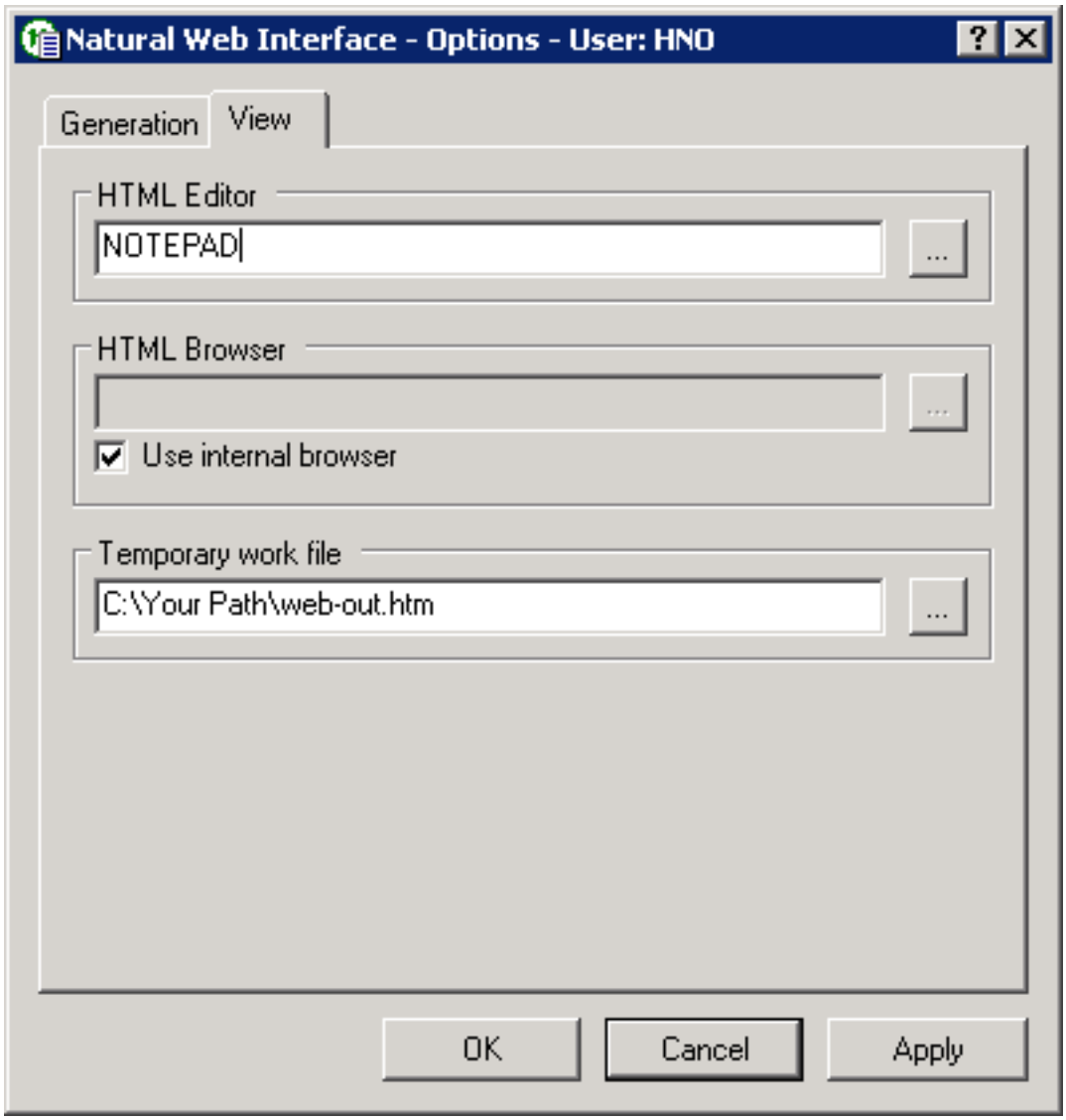


Input/Output Fields

Field	Explanation
Delete unnecessary white space	If checked, multiple white-space characters such as blank, new line, tab, will be reduced to a single white space. For special HTML tags such as `<PRE>` `<TEXTAREA>` or `<SCRIPT>`, the white space will not be collapsed. Default value: unchecked
Save `<NATURAL NOT> ... <NATURAL>` at source	If checked, the content of `<NATURAL NOT>` tags will not usually be generated into the Natural source. This option generates the content of `<NATURAL NOT>` as comment into the Natural source. Default value: unchecked
Stow after generation	If checked, the generated program will be stowed if the generation has been successful. Default value: checked
Use long text constants	Generate text constants longer than 253 characters for better performance. Default value: checked

Field	Explanation
Natural line length	The length of the generated Natural source lines: the minimum value is 20, the maximum 246. Default value: platform dependant
Delimiter	Delimiter string for replacement strings. Default value: $

View



Below is information on:

- Input/Output Fields

- Buttons

Input/Output Fields

Field	Explanation
HTML Editor	The external program that is used to edit the source of the HTML page. Default Value: NOTEPAD
HTML Browser	The external program that is used to display the HTML page. Default: Microsoft Web browser ActiveX Control
Use internal browser	The external program that is used to display the HTML page. To select your own browser, uncheck this box. Default value: checked
Temporary work file	The default output file to be used for displaying data in the HTML-browser. Default value: `C:\Temp\web-out`

Buttons

OK	Leaves the dialog and saves the changes.
Cancel	Leaves the dialog without saving your changes.
Apply	Saves the current input.

Class Generation

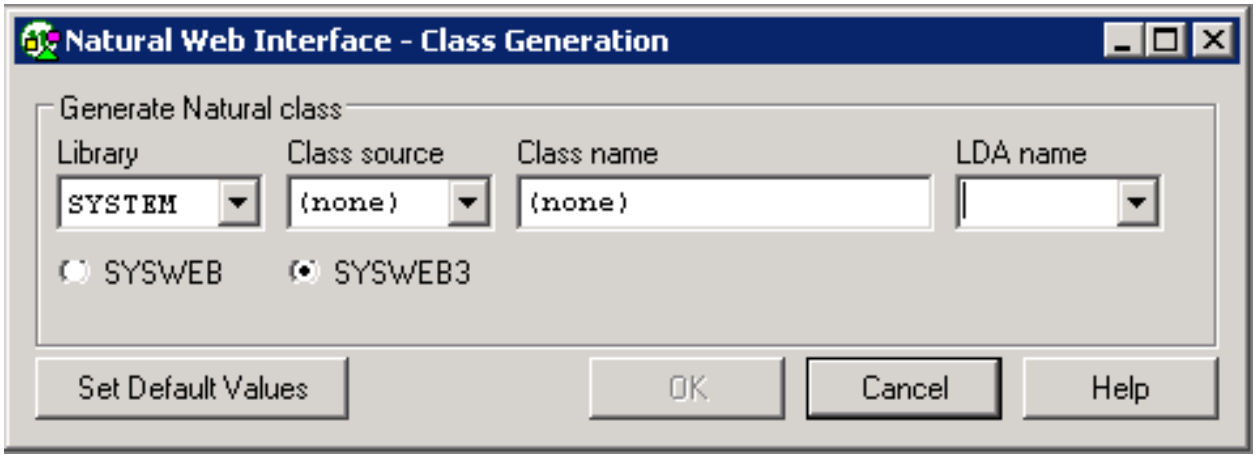
If the Natural Web Interface subprograms should be called using DCOM instead of RPC, a DCOM class is needed. This class contains as methods all relevant Natural subprograms for the Natural Web Interface.

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

Below is information on:

- Input/Output Fields
- Radio Buttons
- Buttons

- Example



Input/Output Fields

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

Radio Buttons

Radio Button	Explanation
SYSWEB	Choose this option if you want to generate a class to be used with library SYSWEB.
SYSWEB3	Choose this option if you want to generate a class to be used with library SYSWEB3.

Buttons

Button	Explanation
OK	Generates the class and leaves the dialog.
Cancel	Leaves the dialog without generation.
Set Default Values	This button is enabled if no relevant class is found for the library. The defaults for Class source and Class name are given. The required LDA has to be generated in advance.

Example

The LDA name is LSYSWEB3. Name the first GUID CLSID- followed by the library name and the second GUID IID-NATWEB3.

```
   T    Comment
        *** Top of Data Area ***
X  U  1 CLSID- SYSWEB3        A 36
X  U  1 IID-NATWEB3           A 36
        *** End of Data Area ***
```

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

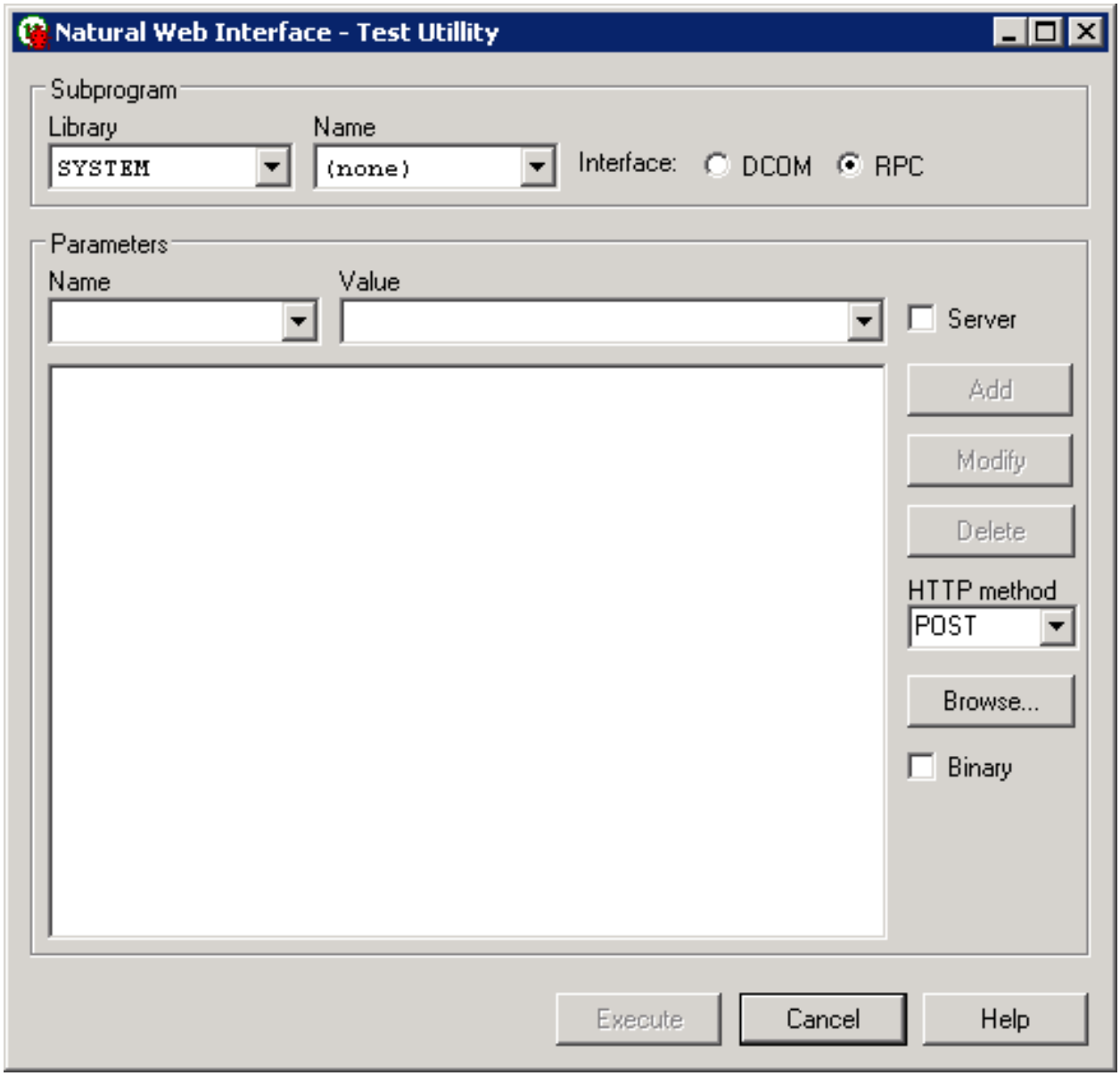
Online Test Utility

This test utility is a component of the Natural Web Interface. You can check your subprogram locally without involving an HTTP server. The transfer parameters for your web page are transferred into the test utility and are posted directly to the business logic.

As communication platform, you can choose either RPC or DCOM as in real remote communications. The result is either the web page expected or an error message. The web page can be viewed with the browser or a viewer of your choice. If you receive an error message, you can easily debug your business logic locally without writing an extra test routine. No remote debugging is necessary.

Features:

- Local application checking.
- No need for remote debugging.
- Simplified error checking.
- Comfortable operation by user friendly interface.
- No need to write an extra test routine.



Below is information on:

- Prerequisites
- Running the Application
- Supported Content Types
- Input/Output Fields

- Buttons:

Prerequisites

- Web browser which supports different content types, for example, Microsoft Internet Explorer Version 5.0 or higher.
- Any available text editor.

Running the Application

To define path adjustments

1. Start the main dialog.
2. Select a browser and viewer of your choice via **Tools > Development Tools > Web Interface Options...**
3. Set the browser, viewer and work file path.
4. Press the OK button.

To start the application

1. Start the dialog WEB-ONL for SYSWEB or WEB-ONL3 for SYSWEB3.
2. Select a library and subprogram name.
3. Optional: add parameters.
4. Choose RPC or DCOM.
5. Press the Execute button.
6. View the result by pressing either the Result... or the Browse button.

Supported Content Types

The following Content Types are supported by the Test Utility:

Content Type	Extension
"application/rtf"	"rtf"
"application/powerpoint"	"ppt"
"application/msword"	"doc"
"application/excel"	"xls"
"text/html"	"htm"
"text/plain"	"txt"
"text/xml"	"xml"

Content Type	Extension
"text/richtext"	"rtf"

If you need further Content Types, change the subroutine HTML2CONTENT-TYPE (SYSWEB/W3CO2EXT or SYSWEB3/W3CO2EXT) and extend the translation table to suit your own needs.

Input/Output Fields

Field	Explanation
Subprogram: Library Name	Enables you to specify the library and the name of the required subprogram. The available libraries and subprograms are automatically taken from the library workspace and listed in selection boxes.
Interface	Can be selected with either DCOM or RPC as communication form. For DCOM, you have to register your classes first. Default: RPC
Parameters: Name Value Server	Here you can enter the name-value-pairs needed from the subprogram. To take them over into the parameter list, press the Add button. To modify the entries, use the Modify button. You do not have to substitute &, =, %; this will be done by the WEB-ONL program. If you use server parameters, check the Server toggle button before you add the parameter to the parameter list. In the parameter list, all name-value-pairs are displayed. &, =, % are substituted. To delete a pair, select the item and press the Delete button. Every selected item will be inserted into the Name and Value fields. If you wish to modify a pair, select the item, change it in the Name and Value fields and press the Modify button. Server: If any of the name-value-pairs are server variables, you need to check this toggle button. Note that any status will last until you change it again.
HTTP Method	In this drop-down list you can select the HTTP request/submit method to be used: ■ **HEAD** Identical to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content. ■ **GET** Requests a representation of the specified resource. ■ **POST** Submits data from the identified resource. The data is included in the body of the request. You can use this method to submit data with a different content type, for example XML files or binary data (such as graphics). If you specify this method an additional **Browse...** button and the **Binary** checkbox are available on the screen.

Field	Explanation
	Use the **Browse...** button to choose a file and the **Binary** checkbox, if you want to submit binary data. If you specify this method without an input file, the default mime type "application/x-www-form-urlencoded" is set. If you use an input file, the content type of that file will be used, for example with an XML file, the content type will automatically be set to "text/xml". You can specify a different mime type in the input field manually. **Note:** A mime type which has been set manually will always override the default mime type. ■ **PUT** Uploads a representation of the specified resource. You can use this method to submit data with a different content type, for example XML files or binary data (such as graphics). If you specify this method, an additional **Browse...** button and the **Binary** checkbox are available on the screen. Use the **Browse...** button to choose a file and the **Binary** checkbox, if you want to submit binary data.

Buttons:

Runs the process of receiving the output from the requested subprogram. The status of the process can be seen in the status bar at the bottom of the WEB-ONL or WEB-ONL3 dialog window.

Starts the editor. It is disabled as long as you have not executed the program and if you have not changed the subprogram library or name. You can choose the editor with the Options dialog.

Starts the browser chosen with the Options dialog. It is disabled as long as you have not executed the program.

V Natural Web Online Documentation SYSWEB3

This section covers the following topics:

About SYSWEB3

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 `SYSWEB3` to display the main page of online documentation at your web browser.

Example of the URL to call the online documentation:
`http://yourserver/yourcgi3/sysweb3/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.

SYSWEB3 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.

Program	Description
`W3BINARY`	Writes **binary data** to the internal binary buffer.
`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** SYSWEB3 and prepares the document for returning to the HTTP server.
`W3LIMIT`	Sets the maximum return page size. See `W3COUNTER` above.
`W3LIST-ENVIRONMENT` `W3LIST-ENVIRONMENT-TO-DYNAMIC`	Lists all **variables**.
`W3LOCATION`	Sets the **location** of a page that is to be called instead this page.
`W3MIME-DATA`	Evaluates the required **data type** (binary or alpha) for the given mime-type.
`W3MIME-TYPE`	Evaluates the file extension for a given **mime-type**. Evaluates the mime-type for a given file extension.
`W3READ-ENVIRONMENT` `W3READ-ENVIRONMENT-ARRAY`	Reads a variable **sent** by the HTTP server.

Program	Description
W3READ-ENVIRONMENT-TO-DYNAMIC	
W3READ-ENVIRONMENT-TEXTAREA W3READ-ENVIRONMENT-TEXTAREA-DYN	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.
W3READ-INPUT	Reads all **data delivered** from the HTTP server.
W3TEXT W3TEXTLINE W3TEXTDYNAMIC W3TEXTLINEDYNAMIC W3TEXTARRAY	Writes a **text string** to the output page.
W3NEWLINE	Writes a **linebreak** to the output page.
W3SOURCE-TO-XARRAY	Reads a complete **Natural source** into one single x-array.
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**.
W3TEXT-TO-URL W3-ASCII-URL-TABLE	Converts ASCII to the specific encoding of **URL**.

SYSWEB3 Template / XSLT Processing

Program	Description
W3REPLACE	Search the output page for a specific string and **replace** with a new one.
W3LOAD-RESOURCE	Load a file from the **resource** directory of a specific Natural library as result document.
W3READ-RESOURCE	Read a file from the **resource** directory of a specific Natural library into a dynamic variable.
W3WRITE-RESOURCE	Write a dynamic variable to a file at the **resource** directory of a specific Natural library.
W3CHECK-RESOURCE	Check if **resource** exits. If not, a new resource can be created.
W3DELETE-RESOURCE	Delete a **resource** from the resource directory.
W3APPLY-XSLT-XML	Apply a **stylesheet** to transformation on the output page.
W3APPLY-XSLT-RESOURCE	Apply a **stylesheet**, saved as resource on the output page.
W3APPLY-XSLT-XML-TO-DYNAMIC	Apply a **stylesheet**, to transformation to an dynamic variable.
W3LIST-RESOURCE	List all **resource** files of a specific Natural library.
W3READ-OUTPUT	Read the **already written** output page.
W3CNTR	Maintain an external **counter**.

Program	Description
W3TAGGER	Process user-defined **tags**.

SYSWEB3 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> <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	Generates a **table**.	<TABLE... >

Program	Description	HTML Tag
H3-TABLE-COLOR		<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

SYSWEB3 Utilities

SYSWEB3 Web Interface Plugin

Plugin	Description
Program Generation	See Natural documentation
Class Generation	See Natural documentation
Online Test Utility	See Natural documentation

SYSWEB3 Online Utilities

Dialog	Description
WEB-ONL or WEB-ONL3	See Natural documentation

SYSWEB3 Remote Utilities

Program	Description
NAT-LIB	Lists **all** Natural libraries.
NAT-DATA	Lists all **header/data** (binary, alpha, unicode?) delivered from the HTTP server.
NAT-DIR	Lists the **contents** of a specific Natural library.
NAT-DIRR	Lists the **resource** contents of a specific Natural library.
NAT-ENV	Lists all parameters **passed** to a called Natural subprogram.

Program	Description
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**.
NAT-RES	Displays a non-Natural file - **resource** (only platform shared resources are available).

SYSWEB3 Demonstration Application

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.

The implementation of the demonstration application uses templates, XML and XSLT, name prefix D6* Depending on your installation of the HTTP Server Extensions, call the subprogram D6INDEX from the library SYSWEB3.

Example of the URL to call the demonstration application:
`http://yourserver/yourcgi/sysweb3/d6index`

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.

20 Writes Binary to the Document

Subroutine Name	Executable Example	Viewable Example
W3BINARY	E3BINARY	E3BINARY

Description

Writes a binary to the document. A document can only contain either a binary or an alphanumeric return value - never both.

Parameters

W3BINARY

```
1 H3BINARY (A) DYNAMIC BY VALUE /* i /mH: Output string
```

How To Invoke

```
PERFORM W3BINARY H3BINARYDYNAMIC
```

21 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
```

22 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
```

23 Count Size of Output Area

Subroutine Name	Executable Example	Viewable Example
W3COUNT W3LIMIT	E3COUNT	E3COUNT

Description

Returns the number of bytes already written there.

Changes from previous versions

New with Natural Version 6.2: The size of the output area is no longer limited by internal settings during compile time. Therefore the maximum size is only limited by the maximum size of a Natural dynamic variable and the parameter settings used for the RPC or DCOM server. The return value of `W3MAXPAGE` can now be set within the program `W3LIMIT`, but has no effect on the maximum output size generated and stored internally.

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
```

24 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
```

25 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"` etc.

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 VALUE /* 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
```

26 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
```

27 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.

Changes from previous version

The LDA `W3PINFO` has been changed, and the variables `LOG-TIME`, `VERSION` and `WEBAPI` have been changed to (A) dynamic.

Parameters

```
LOCAL USING W3PINFO
```

PDA W3PINFO

```
1 ##W3INFO
  2 ACTION   (A1)        /* i /m : Action to be called
  2 LOG-TIME (A) DYNAMIC /*  o/m : Log time set by the Natural Web Interface
  2 VERSION  (A) DYNAMIC /*  o/m : Version set by the HTTP Server Extension
  2 WEBAPI   (A) DYNAMIC /*  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     (A1/1:10)   /* io/m : Saves the type of LIST
  2 LIST_MAX (N2)        /* io/m : Current number of nested LISTs
```

How To Invoke

```
PERFORM W3INFO ##W3INFO
```

28 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 PDA `W3PARM` must be passed to initialize passed parameters for further use. `W3END` ends the document and prepares the return to the HTTPserver. The PDA `W3PARM` 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.

With SYSWEB3, `W3PARM` has been changed to provide an improved interface to the HTTP server. The new interface is capable of transferring binary or alpha data, using other mime-types such as 'application/x-www-form-urlencoded' for incoming data and allows the usage of HTTP PUT requests.

For compatibility reasons, the new features HTTP PUT and other mime-types for data input can only be used, if `W3INIT` is called with an additional parameter:

2nd Parameter := TRUE - activates other mime-types then 'application/x-www-form-urlencoded' for incoming data
3rd Parameter := TRUE - activates HTTP PUT requests.

Changes from previous version

New optional parameters added.

Parameters

1. W3INIT

```
##RPC /* i /m : Parameter of Subprogram
TRUE  /* io/  : TRUE to activate mime-types
TRUE  /* io/  : TRUE to activate HTTP PUT
```

2. W3END

```
##RPC /* o/m : Parameter of Subprogram
```

3. W3PARM

```
1 ##RPC            /* use only with SYSWEB3
  2 LOG-TIME       (A30)       /* i /m : Timestamp
  2 VERSION        (A) DYNAMIC /* i /m : Interface version
  2 HTTP_HEADER    (A) DYNAMIC /* io/m : HTTP header
  2 C_HTTP_HEADER  (I4)        /* io/m : Bytes sent
  2 HTTP_BODY      (A) DYNAMIC /* io/m : HTTP body alphanumeric
  2 HTTP_BINARY    (B) DYNAMIC /* io/m : HTTP body binary
  2 C_HTTP_DATA    (I4)        /* io/m : Bytes sent
  2 MIME-TYPE      (A) DYNAMIC /* io/m : Mime-type sent
  2 ERROR-NR       (I4)        /* io/m : Generation result
```

How To Invoke

```
PERFORM W3INIT ##RPC
PERFORM W3INIT ##RPC TRUE /* activate mime-types ne ↩
'application/x-www-form-urlencoded'
PERFORM W3INIT ##RPC TRUE TRUE /* activate mime-types and HTTP PUT
PERFORM W3END ##RPC
```

29 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)       /* IN/OUT: START FORM
*                               /* out: = 0 :all read
*                               /*      > 0 :data left, reread form here
1 W3ARRAYCOUNTER     (I4)       /* IN/OUT: LENGTH OF THE ARRAY TO RETURN
1 W3ARRAYNAME        (A72/1:V)  /*    OUT: NAME OF THE VARIABLE
1 W3ARRAYVALUE       (A250/1:V) /*    OUT: VALUE OF THE VARIABLE
1 W3ARRAYVALUELENGTH (I4/1:V)   /*    OUT: LEN OF THE VARIABLE
1 W3ARRAYVALUESERVER ( L/1:V)   /*    OUT: Server Variable
1 W3ARRAYMAXIMUM     (I4)       /*    OUT: NUMBER OF THE VALUES
```

2. W3LIST-ENVIRONMENT-TO-DYNAMIC

```
1 W3START            (I4)              /* IN/OUT: START FORM
*                                      /* out: = 0 :all read
*                                     /*      > 0 :data left, reread form here
1 W3ARRAYCOUNTER     (I4)             /* IN/OUT: LENGTH OF THE ARRAY TO RETURN
1 W3ARRAYNAME        (A/1:V) DYNAMIC   /*    OUT: NAME OF THE VARIABLE
1 W3ARRAYVALUE       (A/1:V) DYNAMIC   /*    OUT: VALUE OF THE VARIABLE
1 W3ARRAYVALUESERVER (L/1:V)           /*    OUT: Server Variable
1 W3ARRAYMAXIMUM     (I4)              /*    OUT: NUMBER OF THE VALUES
```

How To Invoke

1.
```
PERFORM W3LIST-ENVIRONMENT W3START W3NAME W3ARRAYCOUNTER
        W3ARRAYNAME(*) W3ARRAYVALUE(*) W3ARRAYVALUELENGTH(*)
        W3ARRAYVALUESERVER(*) W3ARRAYMAXIMUM
```

2.
```
PERFORM W3LIST-ENVIRONMENT-TO-DYNAMIC W3START W3NAME W3ARRAYCOUNTER
        W3ARRAYNAME(*) W3ARRAYVALUE(*) W3ARRAYVALUESERVER(*)
        W3ARRAYMAXIMUM
```

30 Evaluate Mime-Type and Transfer/Data-Type

Subroutine Name	Executable Example	Viewable Example
W3MIME-DATA MIMEDATA	NAT-RES?source=e3xslt2.xml&lib=sysweb3 NAT-RES?source=composing_natural_logo.jpg&lib=sysweb3	NAT-RES

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
```

31 Evaluate Mime-Type and File Extension

Subroutine Name	Executable Example	Viewable Example
W3MIME-TYPE MIMETYPE	NAT-RES?source=e3xslt2.xml&lib=sysweb3 NAT-RES?source=composing_natural_logo.jpg&lib=sysweb3	NAT-RES

Description

Natural Web Interface holds an own table for mime-type / file extension settings. This table is saved in the Natural text object MIMETYPE. The text object contains pairs of mime-type names and file extension(s):

```
text/html     html htm
text/plain    txt
text/xml      xml
text/richtext rtf
```

`W3MIME-TYPE` reads the internal table and offers the following translations:

- Evaluate a mime-type for a given file extension.
- Evaluate a file extension for a given mime-type

The functionality is used internally at the `W3LOAD-RESOURCE` subprogram.

Parameters

1. W3MIME-TYPE

```
W3MIME-TYPE      (A) DYNAMIC  /* i /m : requested mime-type
W3FILE-EXTENTION (A) DYNAMIC  /* o/m : found file extension
```

2. W3MIME-TYPE

```
W3MIME-TYPE      (A) DYNAMIC  /* o/m : found mime-type
W3FILE-EXTENTION (A) DYNAMIC  /* i /m : requested file extension
```

How To Invoke

```
PERFORM W3MIME-TYPE W3FILE-EXTENTION
```

32 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
```

33 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
```

34 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
```

35 Read Environment Text Area Variables

Subroutine Name	Executable Example	Viewable Example
W3READ-ENVIRONMENT-TEXTAREA W3READ-ENVIRONMENT-TEXTAREA-DYN	E3ENVTX E3ENVTXD	E3ENVTX E3ENVTX1 E3ENVTXD E3ENVTX2

Description

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

Parameters

1. W3READ-ENVIRONMENT-TEXTAREA

```
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
```

2. W3READ-ENVIRONMENT-TEXTAREA-DYN

```
1 W3START            (I4)              /* io/m : Offset to be started at
*                                      /*        out: 0 no occurrences
*                                      /*        out: >0 more occurrences
1 W3NAME-DYN         (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 W3ARRAYVALUE-DYN   (A/1:V) DYNAMIC   /*  o/m : Value of variables
1 W3ARRAYVALUELENGTH (I4/1:V)          /*  o/m : Length of variables
1 W3ARRAYMAXIMUM     (I4)              /*  o/m : Total number of variables
```

How To Invoke

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

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

36 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
```

37 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
```

38 Read Natural Source into X-Array of Dynamic

Subroutine Name	Executable Example	Viewable Example
W3SOURCE-TO-XARRAY	E3NAT2XA	E3NAT2XA

Description

Reads a Natural source from a given library into one single x-array.

Parameters

```
1 W3LIBRARY    (A8) DYNAMIC BY VALUE   /* i /o : Natural Library ID
1 W3SOURCE     (A8) DYNAMIC BY VALUE   /* i /m : Source Name
1 W3OUTSRC-XA  (A/1:*) DYNAMIC         /*  o/o : Response Array
1 W3RESPONSE   (I4)                    /*  o/o : Response
```

How To Invoke

```
PERFORM W3SOURCE-TO-XARRAY W3LIBRARY W3SOURCE W3OUTSRC-XA(*) W3RESPONSE
```

39 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
```

40 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 according to the user's needs.

Parameters

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

How To Invoke

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

41 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 according to the user's needs.

Parameters

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

How To Invoke

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

42 Replace Inside Return Document

Subroutine Name	Executable Example	Viewable Example	Resources
W3REPLACE	E3TEMPL	E3TEMPL	E3TEMPL.HTML

Description

Search the already written output page for a specific string and replace all occurrences with a new string. Use together with `W3LOAD-RESOURCE` for template processing.

With the encoding parameter, the given data will be encoded before the replacement is done:

- " " for no encoding
- "HTML" for HTML encoding (e.g. < becomes `<`)
- "URL" for URL encoding
- "XML" for XML encoding (e.g. < becomes `<`)

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"
```

43 Load Resource

Subroutine Name	Executable Example	Viewable Example	Resources
W3LOAD-RESOURCE	E3RESOUR	E3RESOUR	E3SAVE.HTML

Description

Load a file from the resource directory of a given library into the output page.
If no library is specified, use the current library.
Use together with `W3REPLACE` for template processing.

Data is loaded as binary or alpha - depending on the settings of "MIMEDATA" and "MIME-TYPE".
Use `W3MIME-TYPE` to evaluate the mime-type for the given file extension.

Parameters

```
1 W3LIBRARY      (A8) DYNAMIC BY VALUE OPTIONAL     /* i /o : Natural Library ID
1 W3RESOURCE-FILE (A) DYNAMIC BY VALUE               /* i /m : File Name
1 W3RESPONSE      (I4) OPTIONAL                      /*  o/o : Response
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3LOAD-RESOURCE W3LIBRARY W3RESOURCE-FILE

PERFORM W3LOAD-RESOURCE " " "templ.html"

PERFORM W3LOAD-RESOURCE 1X "templ.html"
```

Restriction

Load Resource is only available on Natural for Windows and Linux supporting non-natural files/resources. Internally it uses workfile 12.

44 Read Resource

Subroutine Name	Executable Example	Viewable Example	Resources
W3READ-RESOURCE	E3RESOUR	E3RESOUR	E3SAVE.HTML

Description

Read a resource file from the given library into a dynamic variable. If no library is specified, use the current library.

Use together with W3APPLY-XSLT for template processing.

Parameters

```
1 W3LIBRARY    (A8) BY VALUE OPTIONAL  /* i /o : Natural Library ID
1 W3FILE       (A) DYNAMIC BY VALUE    /* i /m : File Name
1 W3DYN        (A) DYNAMIC             /*  o/m : Read Resource
1 W3RESPONSE   (I4) OPTIONAL           /*  o/o : Response
```

If the W3RESPONSE parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the W3RESPONSE parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3READ-RESOURCE W3LIBRARY W3FILE W3DYN

PERFORM W3READ-RESOURCE W3LIBRARY W3FILE W3DYN W3RESPONSE
```

Restriction

Load Resource is only available on Natural for Windows and Linux supporting non-natural files/resources. Internally it uses workfile 12.

45 Write Resource

Subroutine Name	Executable Example	Viewable Example	Resources
W3WRITE-RESOURCE	E3RESOUR	E3RESOUR	E3SAVE.HTML

Description

Write a resource file to the given library and fill with a given dynamic variable. If no library is specified, use the current library.

Parameters

```
1 W3LIBRARY    (A8) BY VALUE OPTIONAL  /* i /o : Natural Library ID
1 W3FILE       (A) DYNAMIC BY VALUE    /* i /m : File Name
1 W3DYN        (A) DYNAMIC BY VALUE    /* i /m : Read Resource
1 W3RESPONSE   (I4)OPTIONAL            /* o/o : Response
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3WRITE-RESOURCE W3LIBRARY W3FILE W3DYN
PERFORM W3WRITE-RESOURCE W3LIBRARY "Test.txt" W3DYN
PERFORM W3WRITE-RESOURCE W3LIBRARY W3FILE W3DYN W3RESPONSE
```

Restriction

Load Resource is only available on Natural for Windows and Linux supporting non-natural files/resources. Internally it uses workfile 12.

46 Check Resource

Subroutine Name	Executable Example	Viewable Example	Resources
W3CHECK-RESOURCE	E3RESOUR	E3RESOUR	E3SAVE.HTML

Description

Check if a read a resource file exists at the given library. With the create flag, a new empty resource can be created if one does not already exist. If no library is specified, use the current library.

Parameters

```
1 W3LIBRARY    (A8) BY VALUE OPTIONAL  /* i /o : Natural Library ID
1 W3FILE       (A) DYNAMIC BY VALUE    /* i /m : File Name
1 W3CREATE     (L) BY VALUE            /* i /m : If true, create new file
1 W3PATH       (A) DYNAMIC             /*  o/m : Resource path
1 W3RESPONSE   (I4) OPTIONAL           /*  o/o : Response
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3CHECK-RESOURCE W3LIBRARY W3FILE W3CREATE W3PATH
PERFORM W3CHECK-RESOURCE " " 'test.html" TRUE W3PATH
PERFORM W3CHECK-RESOURCE 1x 'test.html" TRUE W3PATH
PERFORM W3CHECK-RESOURCE W3LIBRARY W3FILE W3CREATE W3PATH W3RESOURCE
```

Restriction:

Check Resource is only available on Natural for Windows and Linux supporting non-natural files/resources.

47 Delete Resource

Subroutine Name	Executable Example	Viewable Example	Resources
W3DELETE-RESOURCE	E3RESOUR	E3RESOUR	E3SAVE.HTML

Description

Delete a resource file from the given library. If no library is specified, use the current library.

Parameters

```
1 W3LIBRARY    (A8) BY VALUE OPTIONAL  /* i /o : Natural Library ID
1 W3FILE       (A) DYNAMIC BY VALUE    /* i /m : File Name
1 W3RESPONSE   (I4)OPTIONAL            /*  o/o : Response
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3DELETE-RESOURCE W3LIBRARY W3FILE
PERFORM W3DELETE-RESOURCE " " 'test.html"
PERFORM W3DELETE-RESOURCE 1x 'test.html"
PERFORM W3DELETE-RESOURCE W3LIBRARY W3FILE W3RESPONSE
```

Restriction

Check Resource is only available on Natural for Windows and Linux supporting non-natural files/resources.

48 Apply XSLT Processing on Return Document

Subroutine Name	Executable Example	Viewable Example
W3APPLY-XSLT-XML	E3XSLT	E3XSLT

Description

Call an XSLT processor with the given XML and replace the output page with the generated data.

Parameters

```
1 W3XSL        (A) DYNAMIC BY VALUE  /* i /m : xsl template
1 W3XML        (A) DYNAMIC BY VALUE  /* i /m : xml data
1 W3RESPONSE   (I4) OPTIONAL         /* o/o : result
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3APPLY-XSLT-XML W3XSL W3XML
PERFORM W3APPLY-XSLT-XML W3XSL W3XML W3RESPONSE
```

Restriction

Apply XSLT Processing is only available on versions of Natural for Windows and Linux that contain the User Exit USR 6001P.

49 Apply XSLT Processing on Return Document from Resource

Subroutine Name	Executable Example	Viewable Example	Resource
W3APPLY-XSLT-RESOURCE	E3XSLT2	E3XSLT2	E3XSLT2.XML E3XSLT2.XSL

Description

Call an XSLT processor with the given XSL Resource and replace the output page with the generated data.

Parameters

```
1 W3LIBRARY       (A8) BY VALUE OPTIONAL  /* i /o : Natural Library ID
1 W3RESOURCE-FILE (A) DYNAMIC BY VALUE    /* i /m : File Name
1 W3RESPONSE      (I4) OPTIONAL           /*  o/o : result
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3APPLY-XSLT-RESOURCE W3XSL W3XML
```

Restriction

Apply XSLT Processing is only available on versions of Natural for Windows and Linux that contain the User Exit USR6001P.

50 Load Style Sheet from the Resource Directory and Apply XSLT Processing on Return Document

Subroutine Name	Executable Example	Viewable Example	Used Resource
W3APPLY-XSLT-XML-TO-DYNAMIC	E3XSLT3	E3XSLT3	E3XSLT3.HTML E3XSLT3.XSL E3XSLT3.XML

Description

Call an XSLT processor and transform the output page with a style sheet loaded from the resource directory of the given Natural library. If no library is specified, use the current library.

Parameters

```
1 W3XSLT     (A) DYNAMIC BY VALUE   /* i /m : xslt
1 W3XML      (A) DYNAMIC BY VALUE   /* i /m : xml
1 W3RESULT   (A) DYNAMIC            /*  o/m : result
1 W3RESPONSE (I4) OPTIONAL          /*  o/o : result
```

If the `W3RESPONSE` parameter is not specified, the Web Interface error processing is triggered if an error occurs. If the `W3RESPONSE` parameter is specified, the parameter contains the Natural message number if an error occurred, or zero on success.

How To Invoke

```
PERFORM W3APPLY-XSLT-XML-TO-DYNAMIC W3XSLT W3XML W3RESULT
PERFORM W3APPLY-XSLT-XML-TO-DYNAMIC W3XSLT W3XML W3RESULT W3RESPONSE
```

Restriction

Load Style Sheet and Apply XSLT Processing are available only on versions of Natural for Windows and Linux supporting non-natural files/resources and containing the User Exit USR 6001P. Internally it uses workfile 12.

51 List Resource Files

Subroutine Name	Executable Example	Viewable Example
W3LIST-RESOURCE	nat-dirr	nat-dirr

Description

List all resource files of a specific Natural library. Use `W3PATTERN` for wildcard selection.

Parameters

```
1 W3LIBRARY    (A) DYNAMIC BY VALUE OPTIONAL /* i / : library
1 W3PATTERN    (A) DYNAMIC BY VALUE          /* i /m : selcection pattern
1 W3FILES      (A/1:*) DYNAMIC               /* o/m : array with all file names
1 W3RESPONSE   (I4) OPTIONAL                 /* o/ : response code
```

How To Invoke

```
PERFORM W3LIST-RESOURCE 1x '*' W3FILES(*)
PERFORM W3LIST-RESOURCE 'SYSWEB' '*.HTM' W3FILES(*)
PERFORM W3LIST-RESOURCE 'SYSTEM' '*.BMP' W3FILES(*) W3RESPONSE
```

52 Read Input Page

Subroutine Name	Executable Example	Viewable Example
W3READ-INPUT	E3MULTIP	E3MULTIP NAT-DATA

Description

Read the input data given from the HTTP server into dynamic variables.

`W3BODY` contains alphanumeric data deliverd with "POST/PUT".
`W3BINARY` contains binary data delivered with "POST/PUT".
`W3HEADER` contains all delivered http header.
`W3DATA` contains the data of the QUERY_STRING, and/or data deliverd with POST if the content-type is application/x-www-form-urlencoded.

Parameters

```
1 W3BODY    (A) DYNAMIC           /* o/m : given body
1 W3BINARY  (A) DYNAMIC OPTIONAL  /* o/ : given binary
1 W3HEADER  (A) DYNAMIC OPTIONAL  /* o/ : given header
1 W3DATA    (A) DYNAMIC OPTIONAL  /* o/ : given parameter data
```

How To Invoke

```
PERFORM W3READ-INPUT W3BODY

PERFORM W3READ-INPUT W3BODY 2X W3DATA

PERFORM W3READ-INPUT W3BODY W3BINARY W3HEADER W3DATA
```

53 Read Output Page

Subroutine Name	Executable Example	Viewable Example
W3READ-OUTPUT	E3RESOUR	E3RESOUR

Description

Read into dynamic variables the output page already written.

Changes from previous versions

With Natural Version 6.2 the optional parameter `W3BINARY` has been added.

Parameters

```
1 W3HEADER  (A) DYNAMIC           /* o/m : written header
1 W3BODY    (A) DYNAMIC           /* o/m : written body
1 W3BINARY  (A) DYNAMIC OPTIONAL  /* o/ : written binary
```

How To Invoke

```
PERFORM W3READ-OUTPUT W3HEADER W3BODY

PERFORM W3READ-OUTPUT W3HEADER W3BODY W3BINARY
```

54 Maintain an External Counter

Subroutine Name	Executable Example	Viewable Example
W3CNTR	E3CNTR	E3CNTR

Description

Maintain a counter which is stored on a text resource file. The counter can be read, set to a specific value or increased. If the resource does not yet exist, it is allocated automatically.

Parameters

```
1 W3CNTNAM   (A) DYNAMIC BY VALUE          /* i /m : Resource name, '.txt' is ↩
added automatically.
1 W3CNTVAL   (A) DYNAMIC                   /* io/m : Counter
1 W3OPTION   (A) DYNAMIC BY VALUE OPTIONAL /* i /o : Option; possible values are:
*                                          /*        'INC'   Increment the ↩
counter by 1 (default)
*                                          /*        'READ'  Read the counter
*                                         /*        'RESET' Set the counter to 0
*                                         /*        'SET'   Set the counter to ↩
W3CNTVAL
1 W3RESPONSE (I4)                          /* o /m : Response code; possible ↩
values are:
*                                          /*        1  Resource name not ↩
specified
*                                          /*        2  W3CNTVAL not numeric ↩
for W3OPTION='SET'
*                                         /*        3  Invalid option W3OPTION ↩
specified
```

How To Invoke

```
CALLNAT 'W3CNTR' W3CNTNAM W3CNTVAL W3OPTION W3RESPONSE

CALLNAT 'W3CNTR' 'MY-COUNTER' W3CNTVAL 'INC' W3RESPONSE
```

55 Process User-Defined Tags

Subroutine Name	Executable Example	Viewable Example
W3TAGGER	E3TAGGER	E3TAGGER

Description

Process user-defined tags in any text such as standard text, XML, HTML, program source etc. In special, it can be used to create a variable table in a template HTML file.

The dynamic alpha variable `W3TEXT` is inspected for the tag named `W3NAME`. All characters between the corresponding start and end tag (the 'tag content') are returned in `W3VALUE`. The option field `W3OPTION` specifies whether the current tag content should be kept or if the value `W3NEWVAL` should be added. The `W3TAG-OPTION` option specifies whether the start and end tag should be deleted from `W3TEXT` or not.

Usage

1. Create a template member with any tool, for example an HTML file with Microsoft Expression Web.
2. Add user-defined tags around the parts which you want to process further. For example around a table row:

   ```
   <MY-TAG><tr><td>$NAME$</td></tr></MY-TAG>
   ```

3. Read the template into a Natural A-dynamic variable with the Natural Web module `W3READ-RESOURCE`.
4. Process the user-defined tags in the template with `W3TAGGER`.
5. Write the modified template to the output page with the Natural Web module `W3TEXTDYNAMIC`.

Parameters

```
1 W3TEXT       (A) DYNAMIC                   /* io/m : Text which contains the ↩
user-defined tags.
1 W3VALUE      (A) DYNAMIC                    /*  o/m : Tag content
1 W3NAME       (A) DYNAMIC BY VALUE           /* i /m : Tag name
1 W3TAG-OPTION (A) DYNAMIC BY VALUE OPTIONAL /* i /o : Tag-Option; possible ↩
values are:
*                                            /*        'DELETE-TAG' Delete the ↩
tag from the text (default)
*                                             /*        'KEEP-TAG'   Keep the ↩
tag in the text
1 W3OPTION     (A) DYNAMIC BY VALUE OPTIONAL /* i /o : Option; possible values ↩
are:
*                                            /*        'DELETE'     Delete the ↩
tag content from the text (default)
*                                             /*        'KEEP'       Keep the ↩
tag content in the text
*                                           /*        'REPLACE'    Replace the ↩
tag content with W3NEWVAL
*                                             /*        'INSERT'     Insert ↩
W3NEWVAL in front of the tag content
*                                          /*        'ADD'        Add W3NEWVAL ↩
behind the tag content
1 W3NEWVAL     (A) DYNAMIC BY VALUE OPTIONAL /* i /o : New tag content for ↩
W3OPTION='REPLACE', 'INSERT' or 'ADD'
1 W3MSG        (A) DYNAMIC                    /* io/m : Message. New messages ↩
are added to the end of W3MSG.
1 W3RESPONSE   (I4)                          /*  o/m : Response code; possible ↩
values are:
*                                             /*        0  ok
*                                           /*        1  Invalid option W3OPTION
*                                             /*        2  Tag not found
*                                             /*        3  End tag not found
*                                          /*        4  End tag before start tag
*                                          /*        5  No input text (W3TEXT) ↩
given
*                                            /*        6  No tag name (W3NAME) ↩
given
*                                             /*        7  Invalid tag option ↩
(W3TAG-OPTION)
```

How To Invoke

```
CALLNAT 'W3TAGGER' W3TEXT W3VALUE W3NAME W3TAG-OPTION W3OPTION W3NEWVAL W3MSG ↩
W3RESPONSE

CALLNAT 'W3TAGGER' W3TEXT W3VALUE 'MY-TAG' 'KEEP-TAG' 'DELETE' 1X W3MSG W3RESPONSE
```

56 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
```

57 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
```

58 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
*                                   /*        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
```

59 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
```

60 Level and 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
```

61 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 of 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
```

62 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
```

63 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. You are recommended to 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
```

64 Form

Subprogram Name	Source Name	Executable Example	Viewable Example
H3-OPEN-FORM H3-CLOSE-FORM	H3OFORM H3CFORM	E3FORM E3FORM2(utf-8)	E3FORM E3FORM2

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
```

65 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="#RPG">
```

3. H3-CLOSE-HTML

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

Supported Attributes

```
BACKGROUND="URL", BGCOLOR="#RPG", 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
```

66 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
```

67 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
```

68 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 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
```

69 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
```

70 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"> </SELECT> <OPTION> </OPTION>
```

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
```

71 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>
<TH>
<TD> ... </TD>
</TH>
<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 (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
```

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
```

72 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
```

73 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 lines
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
```

74 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 according to 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
```

75 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 recognized 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
```

76 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
```

77 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.

78 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. Under Windows, use the **Web Interface Plug-In** .

Input Map

```
12:12:40          ***** Natural Web Subprogram Wizard *****         2005-11-15
                                - Main Menu -                Library SYSWEB



 Subprogram Name ..... DUMMY_
 Title ............... HTTP/HTML API WIZARD_____________________________________
 Header .............. HTTP/HTML API WIZARD_____________________________________

 Use HTML extension .. X









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

How To Invoke

Run Program `WEB-WIZ` from the Natural NEXT prompt.

79 List All Data Passed From a HTTP Server to a Called Natural Subprogram

Subprogram Name	Executable Example	Viewable Example
NAT-DATA	NAT-DATA NAT-DATA?parm1=test1&parm2=test2 E3MULTIP	NAT-DATA E3MULTIP

Description

Generates an HTML page with all data passed from the HTTP server to a called Natural subprogram, including mime-type, HTTP header, parameters, alphanumeric data and binary data.

Parameters

```
/* none
```

How To Invoke

```
NAT-DATA
```

80 List Directory of a Natural Library

Subprogram Name	Executable Example	Viewable Example
NAT-DIR	NAT-DIR?lib=sysweb NAT-DIR?lib=sysweb3&version=no&start=E3* 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.

New with Natural Version 6.2

Version check for the subprogram modules added. Displays SYSWEB if cataloged with library SYSWEB and SYSWEB3 if cataloged with library SYSWEB3. If NAT-DIR is called with the SYSWEB interface then no run link 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.
`VERSION=`	If set to NO, subprograms will not be checked on the SYSWEB version used during runtime (either SYSWEB or SYSWEB3). Default is YES.

How To Invoke

```
nat-dir?lib=sysweb3
```

81 List Resources of a Natural Library

Subprogram Name	Executable Example	Viewable Example
NAT-DIRR	NAT-DIRR?lib=sysweb3 NAT-DIRR?lib=sysweb3&start=*.jpg NAT-DIRR?lib=sysweb&start=*.htm	NAT-DIRR

Description

Generates an HTML page with the resouce file 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-dirr?lib=sysweb3
```

82 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
```

83 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
```

84 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
```

85 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
```

86 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
```

87 List non-Natural File - Resource

Subprogram Name	Executable Example	Viewable Example
NAT-RES	NAT-RES?source=e3xslt2.xml&lib=sysweb NAT-RES?source=composing_natural_logo.jpg&lib=sysweb3 NAT-RES/sysweb3/e3put.html	NAT-RES

Restriction

Only available on platforms with shared resources.

Description

Generates an HTML page with the listing of a Natural source object. Depending on the definition at the text object **MIMEDATA** the data will be transferred as binary or as alphanumeric data.

Parameters

`LIB=`	Natural LIBRARY
`SOURCE=`	Resource name with extension

How To Invoke

```
nat-res?source=e3xslt2.xml&lib=sysweb3
```

How To Invoke with extended URL syntax

Instead of specifying the parameters `LIB=` and `SOURCE=`, library and source name can be added directly after the program name. `nat-res/<yourlibrary>/<yourresource>`

Extended Functionality

If the HTTP server is PUT enabled, NAT-RES can be used to write back to the HTTP server with an HTTP put request.

To activate the PUT capability, modify the subprogram NAT-RES by setting the variable F_PUT to TRUE and recatalog the subprogram NAT-RES.

VI XML Toolkit Plug-In

The XML Toolkit plug-in enables developers to process XML documents within Natural.

The toolkit includes a wizard which generates Natural source code and provides the following features:

- Mapping Natural data definitions to DTDs or XML schemas;
- Serializing a Natural data structure and assigning its contents to an XML file;
- Mapping DTDs to Natural data definitions;
- Parsing an XML file and assigning its contents to a Natural data structure.

The wizard is included in the library SYSXTK.

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

The following topics are covered:

Introduction
Using the XML Toolkit
Setting Up Specific Generation Options
Using a Natural Data Source
Using an external Data Source
Natural Simple XML Parser
Examples
XML Parser Error Messages

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

88 Introduction to the XML Toolkit Plug-In

- About the XML Toolkit Plug-In 278
- XML Toolkit Features 278
- XML Toolkit Description 278
- Considerations and Limitations 283

The following topics are covered:

About the XML Toolkit Plug-In

The XML Toolkit plug-in enables developers to process XML documents within Natural.

The toolkit includes a wizard which generates Natural source code and provides the following features:

- Mapping Natural data definitions to DTDs or XML schemas;
- Serializing a Natural data structure and assigning its contents to an XML file;
- Mapping DTDs to Natural data definitions;
- Parsing an XML file and assigning its contents to a Natural data structure.

The wizard is included in the library SYSXTK.

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 is implemented as a Natural plug-in. 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 or XML Schema 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 or XML Schema 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: SYSXTK  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 A dynamic, the current maximum.

```
* DTD E:\SAG\nat\v.r\fnat\SYSXTK\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\v.r\fnat\SYSXTK\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&

...
```

Considerations and Limitations

The XML Toolkit only supports fully assembled XML Schema (Layer 1). For detailed information, refer to the W3C recommendation on **XML Schema (Layer 1)**

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

- Very Large Data Structures
- Structures that Require Many Array Dimensions
- Multi-Dimensional Arrays
- XML Schema: Access and Composition
- DTD: Add external parsed data
- Conditional DTD
- Wildcards
- Restrictions for Unnamed Groups with Multiple Occurrences

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.

Structures that Require Many Array Dimensions

The XML toolkit uses Natural arrays to map XML schemas that contain elements with multiplicity. Natural arrays are limited to three dimensions. If a schema contains more than three nested levels of multiplicity, try to split off a sub-schema, using the type `xsd:anyType` (see the example below). Both parts can then be processed separately by the XML toolkit.

Example

Original schema with more than three nested levels of multiplicity:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:element name="Document" type="Document"/>

    <xs:complexType name="Document">
        <xs:sequence>
            <xs:element name="a" type="aType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="aType">
        <xs:sequence>
            <xs:element name="b" type="bType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="bType">
        <xs:sequence>
            <xs:element name="c" type="cType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="cType">
        <xs:sequence>
            <xs:element name="d" type="dType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="dType">
        <xs:sequence>
            <xs:element name="e" type="eType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="eType">
        <xs:sequence>
            <xs:element name="f" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Changed schema - part one:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:element name="Document" type="Document"/>

    <xs:complexType name="Document">
        <xs:sequence>
            <xs:element name="a" type="aType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="aType">
        <xs:sequence>
            <xs:element name="b" type="xs:anyType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Changed schema - part two:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:element name="c" type="cType"/>

    <xs:complexType name="cType">
        <xs:sequence>
            <xs:element name="d" type="dType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="dType">
        <xs:sequence>
            <xs:element name="e" type="eType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="eType">
        <xs:sequence>
            <xs:element name="f" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Multi-Dimensional Arrays

The following limitations apply, when generating an XML document from a **Natural data area**, if the source data contains arrays:

- For each array, exactly one dimension is allowed.
- For each level, exactly one dimension can be added.
- Each array must have a counter variable.
- The counter variable
  - must be located before the array and
  - the counter variable name must start with the character C followed by the **counter separator field**.
- The *LBOUND (lower boundary) of the array must be 1.

XML Schema: Access and Composition

<include>
Include adds multiple schemas with the same target namespace to a document. The document needs to be included, without any changes.

<import>
Import adds multiple schemas with different target namespaces to a document. First the document to be imported requires a namespace prefix translation, then the document can be included.

<redefine>
Redefine selects out specific simple and complex types, groups, and attribute groups from an external schema, and enables you to modify the given specification for your own needs.

Note: With all of the above elements, only relative URIs are allowed. Absolute URIs (e.g. *http://www.yourdomain.com/your/path* or *file://your/path*) can not be used.

DTD: Add external parsed data

The external data has to be included into the document. There are no conversions necessary.

Conditional DTD

If a `<![ INCLUDE ]` is found, the contained definition will be used for generation.

If a `<![ IGNORE ]` is found, the contained definition will not be used for generation.

Wildcards

The XML Toolkit supports two different kinds of wildcard representations:

- Save all subsequent elements or
- Save all attributes that are not specified

For wildcard support the following rules and/or limitations apply.

XML Schema <anyAttribute>

For attributes an `attributes_of_<entity-name>` group is generated. All attributes connected to this group are added. The name of an attribute is saved as a variable name; the content is the content of the variable.

To add `<any>` attributes, it is required to add a variable that contains all attributes not specified.

Note: The `<any>` attributes are no "real" attributes; they are used as a container for the not parsed data and contain the attribute/value pairs. An `<any>` attribute is represented by a `##ANY` variable of type (A) dynamic.

Because it can be necessary to access this data, a more specific name should be used instead of `##ANY` followed by a generic number. It is recommended to add the name of the parent entity and the keyword `ATTR`, or `ATTRIBUTE`. See below for an example:

```
1 HTML
  2 BODY
    3 ATTRIBUTES_OF_BODY
      4 BGCOLOR         (A) DYNAMIC
      4 ##ANY_ATTR_BODY (A) DYNAMIC
```

If, during the parse process, an attribute that is not named inside the XML Schema is found, the variable name and value will be saved at the `all_attributes_of_<element-name>` group as they are, this means with the standard XML syntax:

`<attribute-name1>="<attribute-value1>" <attribute-name2>="<attribute-value2>"` etc.

When serializing, the above string will be added.

XML Schema <any> or DTD <!ELEMENT element-name ANY>

To add the `<any>` data type, it is necessary to save all subsequent data of an entity, regardless of the names and values of this entity.

Note: The `<any>` entities do not specify "real" entities; they are used as a container for the not parsed data and contain the entities with their entire content (attributes, etc.). An `<any>` entity is represented by a `##ANY` variable of type (A) dynamic.

Because it can be necessary to access this data, a more specific name should be used instead of `##ANY` followed by a generic number. It is recommended to add the name of the parent entity. See below for an example:

```
1 HTML
  2 BODY
    3 ATTRIBUTES_OF_BODY
      4 BGCOLOR         (A) DYNAMIC
      4 ##ANY_ATTR_BODY (A) DYNAMIC
        3 ##ANY_BODY        (A) DYNAMIC
```

If, during the parse process, an element of type `<any>` is found, all subsequent data is collected.

When serializing, all data is taken without changes and is added to the resulting XML document.

Restrictions for xs:any

Even if the attributes `"maxOccurs"` and/or `"minOccurs"` for `xs:any` are specified, the Natural variable implementing `xs:any` is always a scalar. The Natural variable may contain data of more then one entity.

The attribute `"namespace"` for `xs:any` is not evaluated, the Natural variable implementing `xs:any` may contain entities of different namespaces.

The attribute `"processContents"` for `xs:any` is not evaluated, because the parser used is not validating.

If a `xs:choice` or `xs:sequence` contains more then one definition of `xs:any`, the generation ends with an error, because during parse different `<any>` containers can not be recognized.

If a document contains entities that are not specified at the XML schema, and at the same level `xs:any` is defined, the Natural variable implementing `xs:any` may contain this "nonspecified" entity data.

Natural: Generation of an XML Schema or DTD with ##ANY Wildcards

During generation of an external data structure, each variable prefixed with `##ANY` will be converted to the specific wildcard functions:

- `##ANY_` -> any entity type. Applies to DTDs and XML Schema.
- `##ANY_ATTR_` -> any attribute type. Applies to XML Schema only.

Restrictions for Unnamed Groups with Multiple Occurrences

When multiple occurrences (`maxOccurs>1`) for unnamed group structures like `xs:sequence`, `xs:choice` or `xs:all` are defined then the generated source code might need some manual adaptations. This becomes necessary because unnamed groups usually do not result in unique xpath names that can be used for parsing.

89 Using the XML Toolkit

- Invoking the Application 292
- Getting Help 292

The following topics are covered:

Invoking the Application

The XML toolkit is included in the library SYSXTK.

To use the XML Toolkit

- Select the library SYSXTK from the library workspace.

  In the Dialogs folder choose dialog Menu.

  Or:

  In the Natural command line, enter LOGON SYSXTK.

  Enter Menu.

  The first screen of the dialog wizard is displayed.

  The following functions are available:

  - **Generate from Natural Data Structure** Uses the Natural Data Area as a data source.
  - **Generate from Document Type Definition/XML Schema** Uses the Document Type Definition (.dtd), XML Schema (.xsd) or Tamino Schema 2 (.tsd) as a data source.
  - **Set up Specific Generation Options**

Getting Help

To get help with the XML Toolkit

- In the XML Toolkit main dialog choose XML Toolkit Help from the Help menu.

90 Setting up Specific Generation Options

- Invoking the Natural XML Options Menu 294
- Generation 294
- Path 297
- Saving Your Options Permanently 299

The generation options are arranged on three different tabs and are grouped into generation, view and path definitions.

The following topics are covered below:

Invoking the Natural XML Options Menu

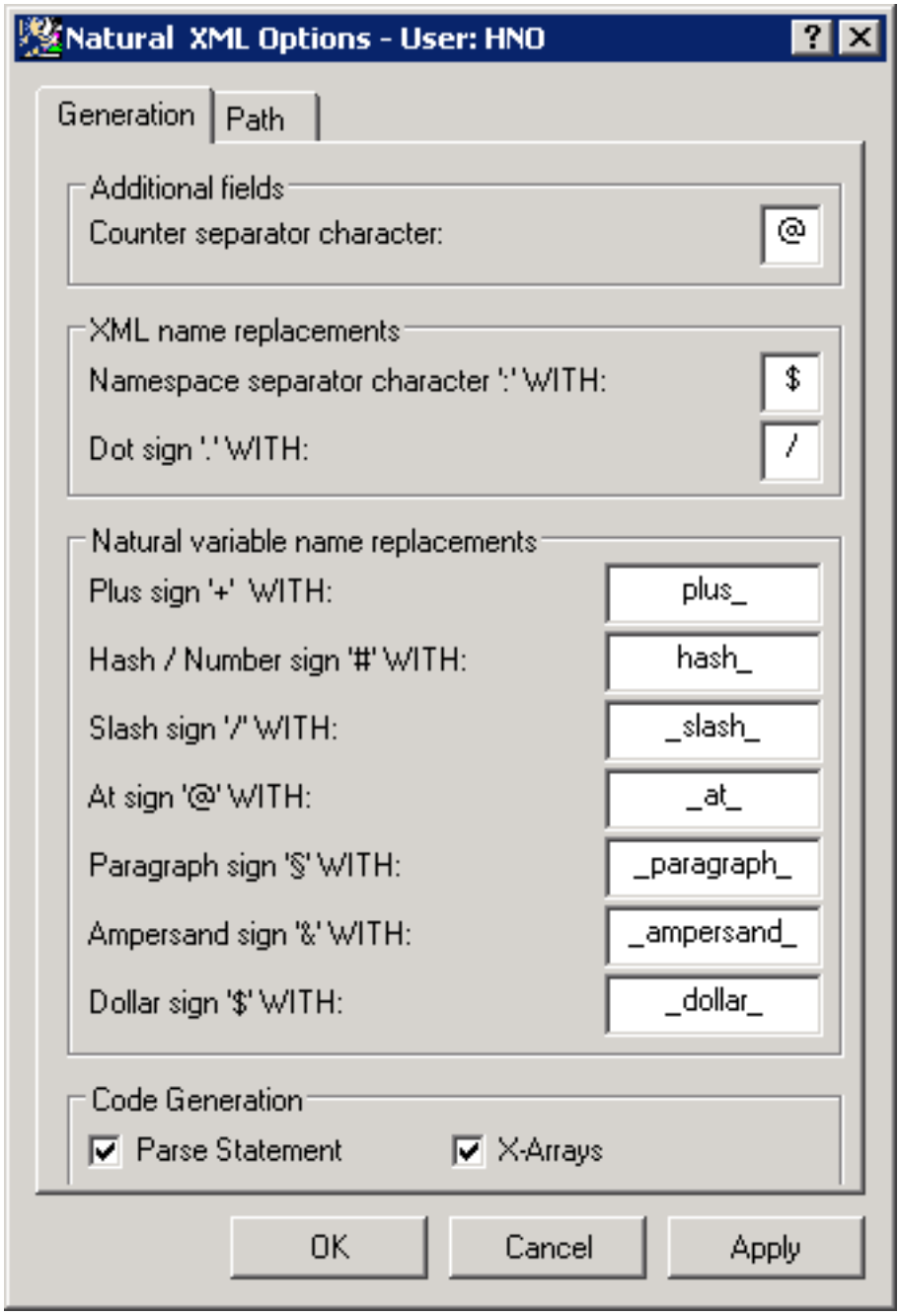
To invoke the Natural XML Options Menu

- Choose Options from the Generate menu.

  The Generation tab of the Natural XML Options menu is displayed. The fields of the menu tabs are described below.

Generation

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



Field Descriptions

Counter Separator Character

Belongs to Group:	Additional Fields
Default Value:	@

Namespace Separator Character ":" WITH:

Belongs to Group:	XML Name Replacements
Default Value:	$

Dot Sign '.' WITH:

Belongs to Group:	XML Name Replacements
Default Value:	/

Plus Sign '+' WITH:

Belongs to Group:	XML Variable Name Replacements
Default Value:	plus

Hash / Number Sign '#' WITH:

Belongs to Group:	Natural Variable Name Replacements
Default Value:	hash

Slash Sign '/' WITH:

Belongs to Group:	Natural Variable Name Replacements
Default Value:	slash

At Sign '@' WITH:

Belongs to Group:	Natural Variable Name Replacements
Default Value:	at

Paragraph Sign '§' WITH:

Belongs to Group:	Natural Variable Name Replacements
Default Value:	para

Ampersand Sign '&' WITH:

Belongs to Group:	Natural Variable Name Replacements
Default Value:	amp

Dollar Sign '$' WITH:

Belongs to Group:	Natural Variable Name Replacements
Default Value:	dollar

Parse Statement

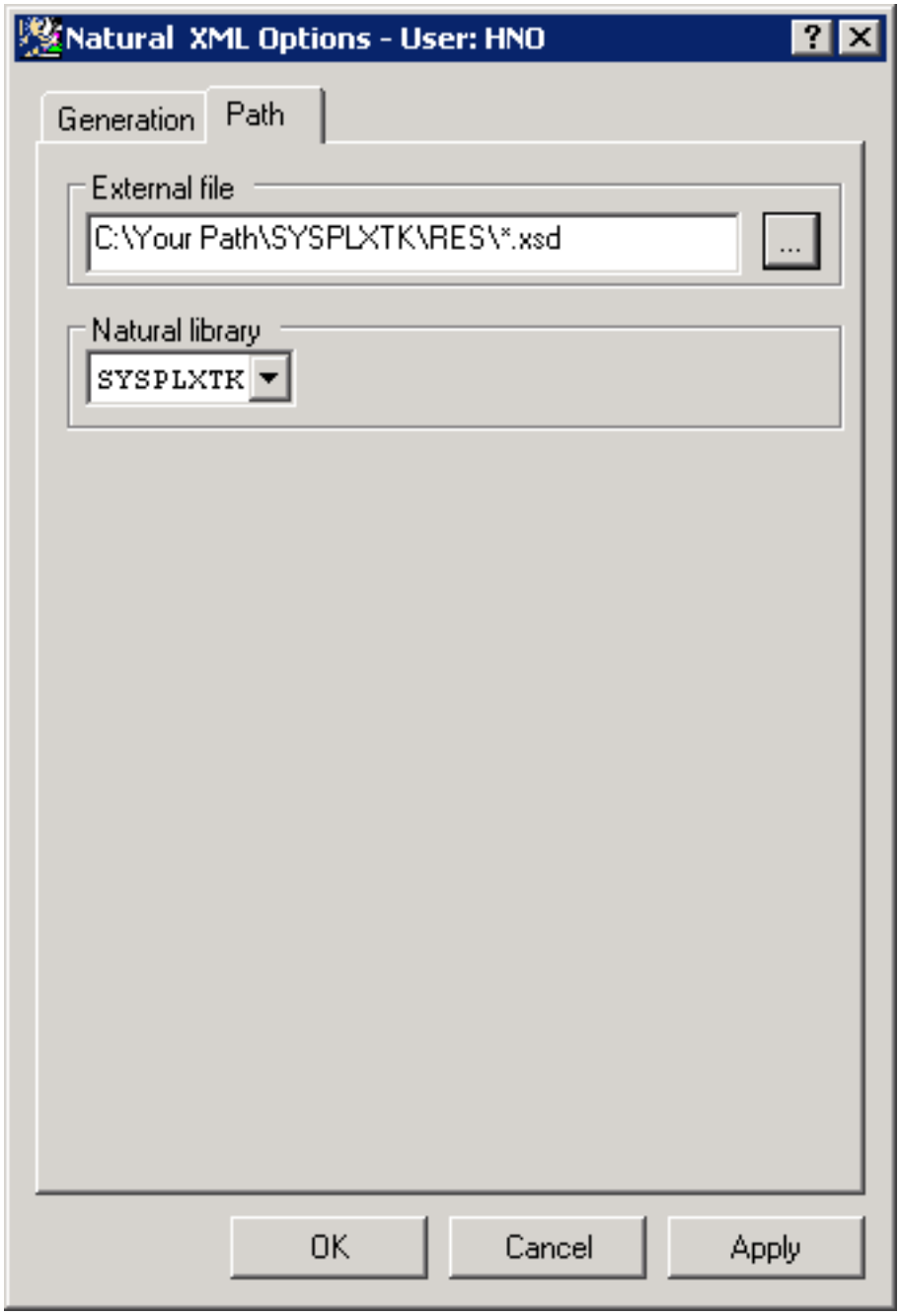
Belongs to Group:	Code Generation
Default Value:	checked (Yes)

X-Arrays

Belongs to Group:	Code Generation
Default Value:	checked (Yes)

Path

The Path tab of the Natural XML Options menu serves to define the location of the target or source DTD file used for the conversion.



Field Descriptions

External file

Default Value:	(Resource directory of current library)

Natural library

Default Value:	(current library)

Saving Your Options Permanently

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

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

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

91 Using a Natural Data Source

- Select Natural Data Area 302
- Select Data Type 304
- Generate File with DTD Definition or XML Schema 305
- Generate a serializer for an XML document 306
- Generate a parser for an XML document 307
- Parameter Settings 309
- Select Root Group 309
- Show Generation Report 310

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**

Select Natural Data Area

This dialog serves to select generation from a Natural Data Structure or a XML Schema or Document Type Definition.

To invoke the dialog shown below

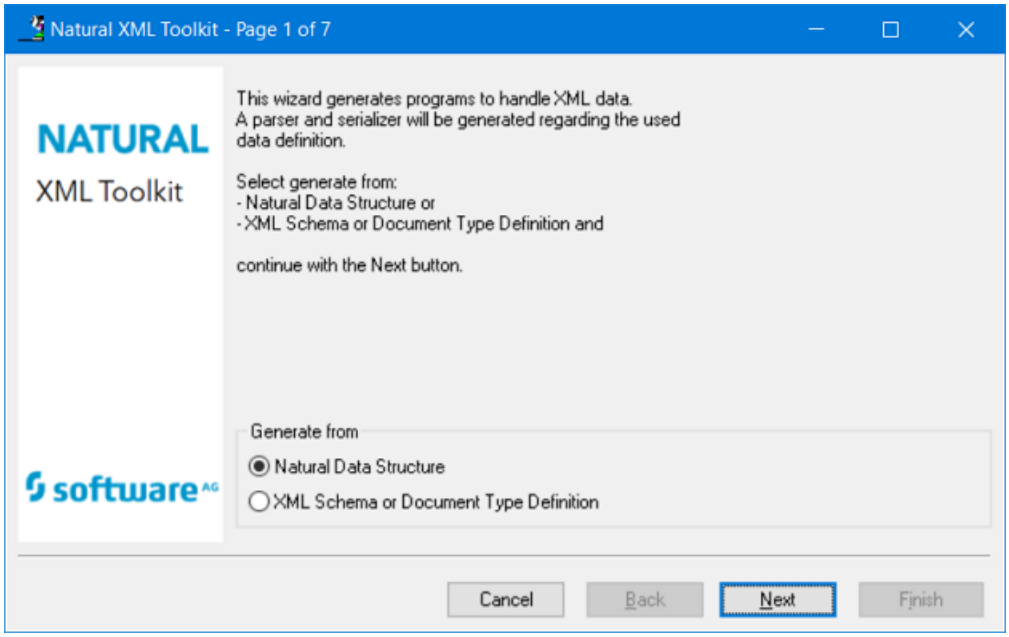
1 Activate the XML Plug-In in 'Tools' > 'Configuration Tools' > 'Plug-In Manager' > 'XML Toolkit'.

Or:

Alternatively choose the button to open the Plug-In Manager.

2 Select 'Tools' > 'Development Tools' > ' XML Toolkit'.

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



Choose if you would like to generate from a Natural Data Source or from a XML Schema or DTD.

Select **Next** to continue.

Field Descriptions

Library

Belongs to Group:	Select Input Data Area
Default Value:	(All libraries)

Type

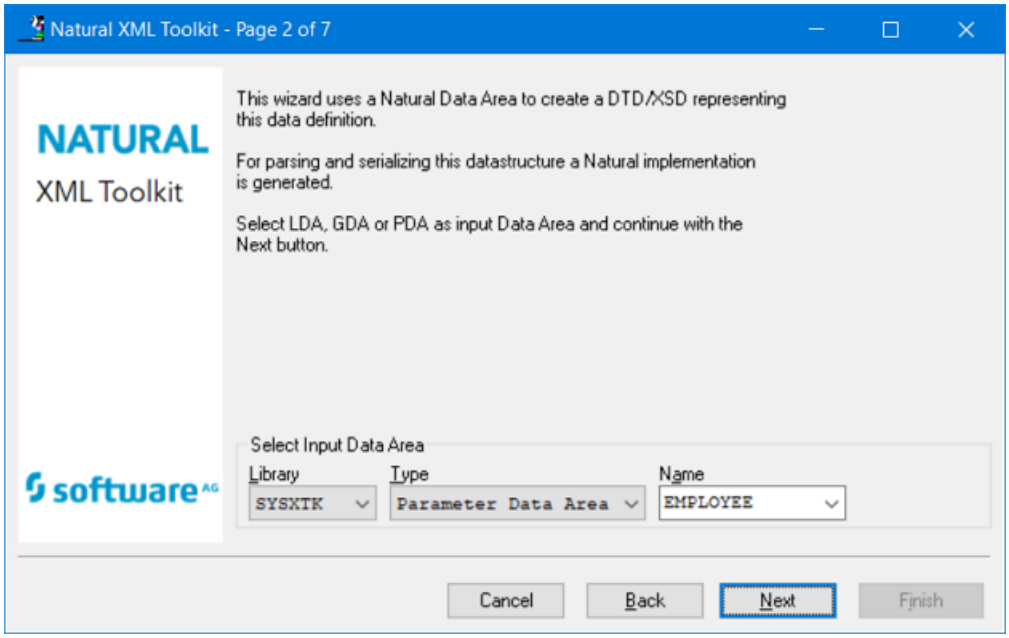
Belongs to Group:	Select Input Data Area

Name

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

Select Data Type

This dialog is used to select the data type.



Field Descriptions

Library

Belongs to Group:	Library
Default Value:	(All libraries)

Type

Belongs to Group:	Type
Possible Values:	Local Data Area Parameter Data Area Global Data Area

Name

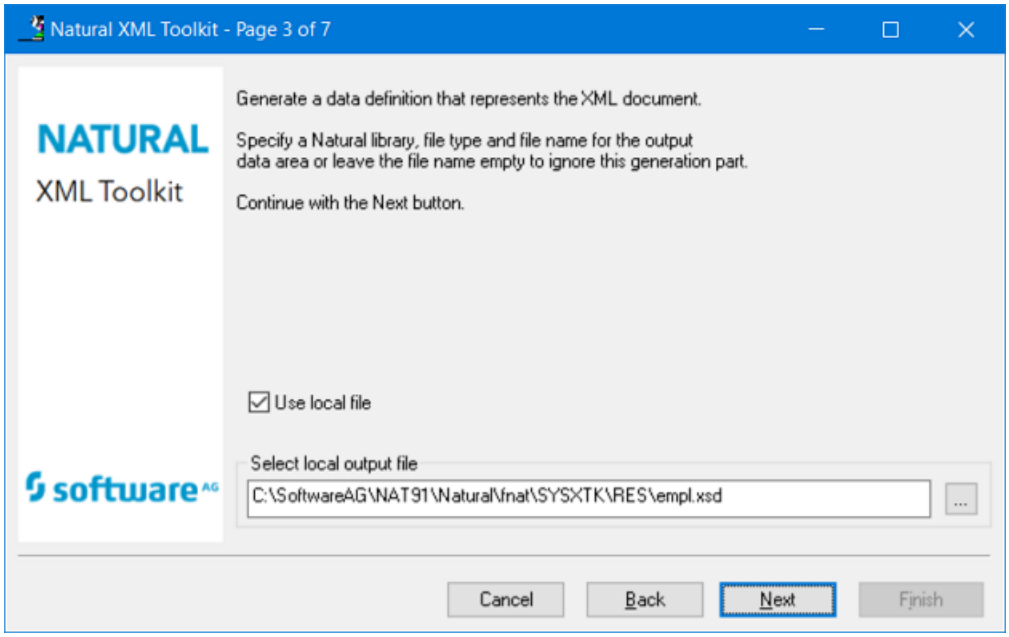
Belongs to Group:	Name
Default Value:	(All objects of the selected library and type)

Select the desired element, e.g. EMPLOYEE .

Choose **Next** to continue.

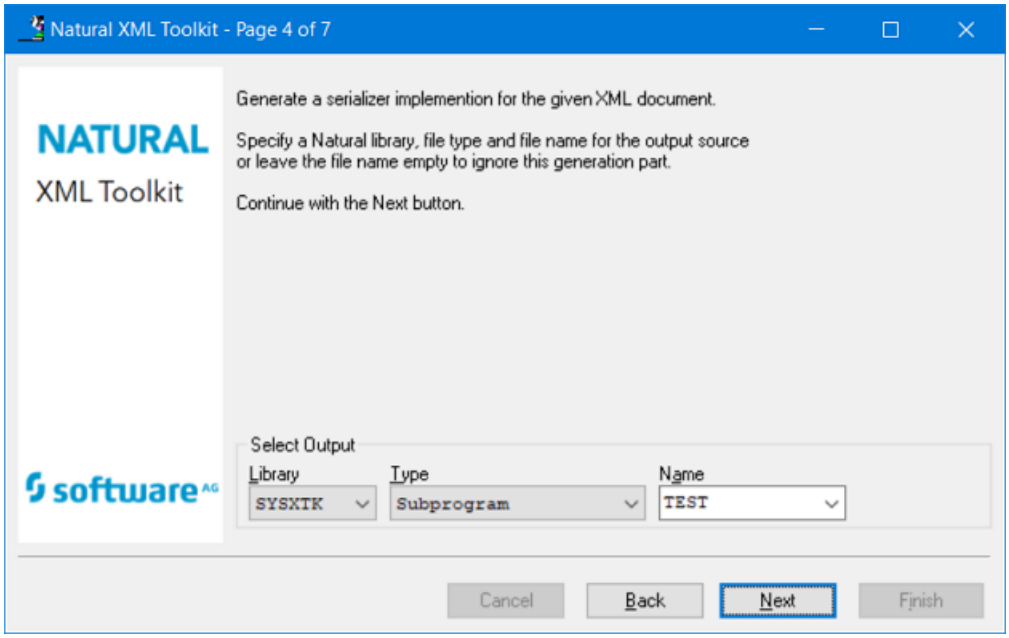
Generate File with DTD Definition or XML Schema

In this dialog you can specify a file name for the output data area.



Choose **Next** to continue.

Generate a serializer for an XML document



This dialog is used to specify a Natural Library, file type and file name for the output source.

Field Descriptions

Library

Belongs to Group:	Library
Default Value:	(All libraries)

Type

Belongs to Group:	Type
Possible Values:	Copycode Subprogram

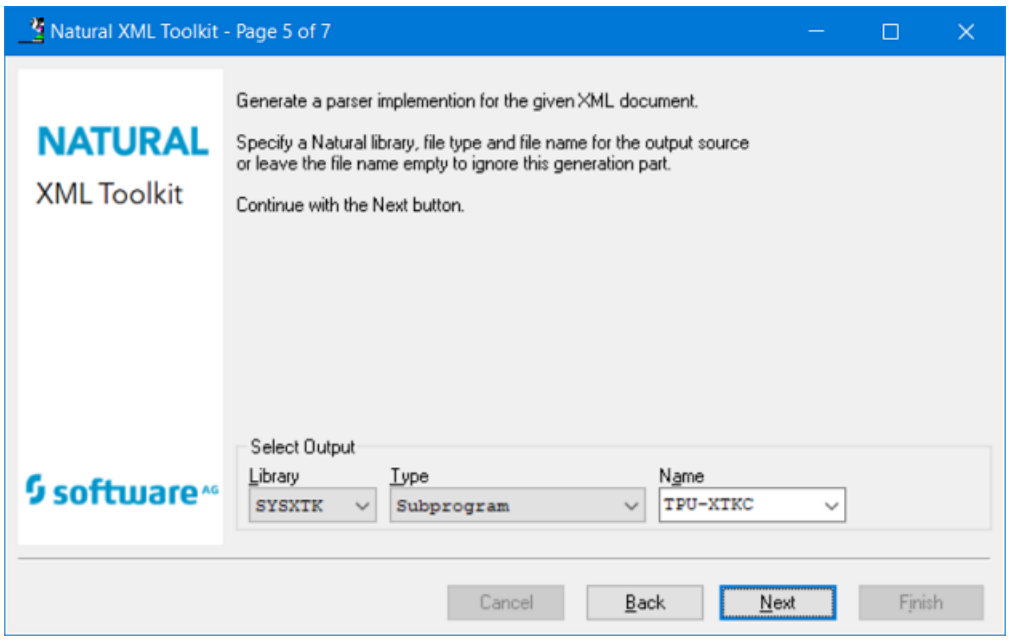
Name

Belongs to Group:	Name
Default Value:	(All objects of the selected library and type)

Choose **Next** to continue.

Generate a parser for an XML document

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



Field Descriptions

Library

Belongs to Group:	Library
Default Value:	(All libraries)

Type

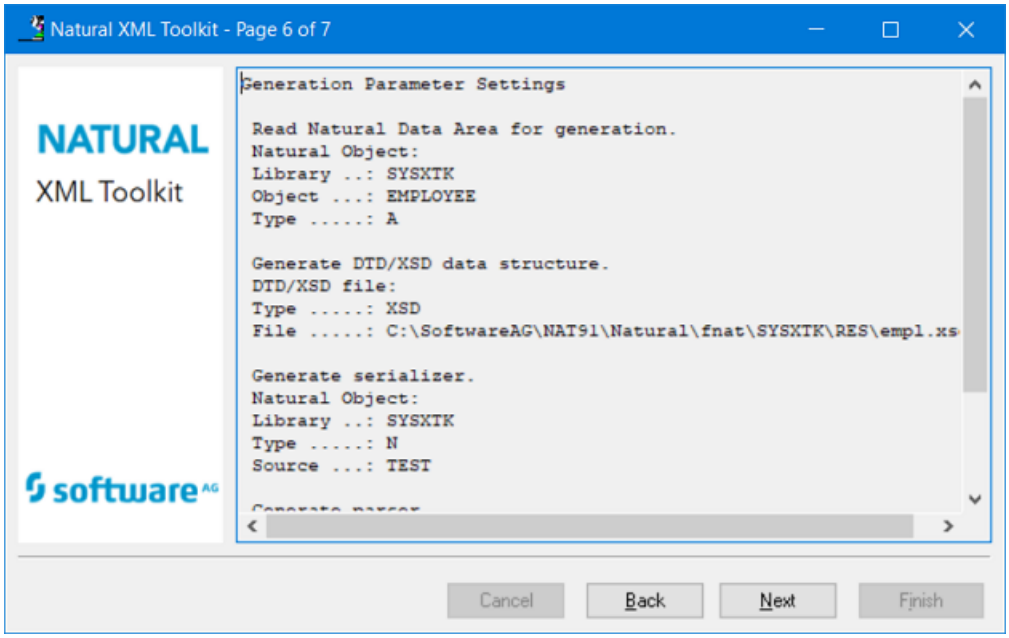
Belongs to Group:	Type
Possible Values:	Copycode Subprogram

Name

Belongs to Group:	Name
Default Value:	(All objects of the selected library and type)

Choose **Next** to continue.

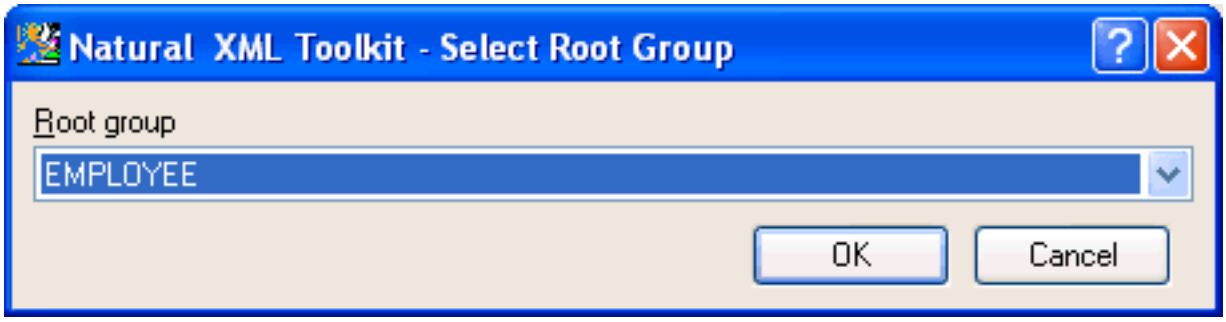
Parameter Settings



This screen shows you the settings used for the generation process.

Choose **Next** to continue.

Select Root Group

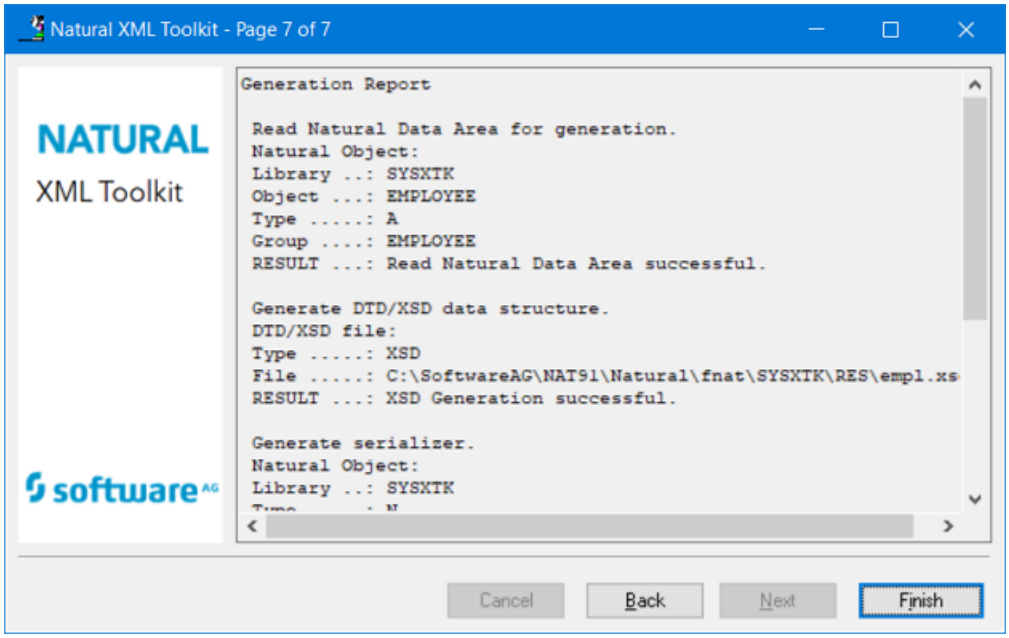


In this dialog you can select the Root Group .

Choose **OK** to continue.

Show Generation Report

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



Choose **Finish** to end the generation process.

92 Using an external Data Source

- Generate from Document Type Definition or XML Schema 312
- Select Root Element or Document Type 313
- Select Recursion Level 314
- Generate Natural Data Area 315
- Generate Copycode for Serialization 317
- Generate Subprogram for Serialization 318
- Generate Copycode for XML Parser Callback 318
- Generate Subprogram for XML Parser Callback 320
- Show Generation Results 320

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**

Note: When using an XML Schema (XSD) as input document type, the first XSD element will be used as the root element.

Generate from Document Type Definition or XML Schema

This dialog is used to select a Document Type Definition (DTD), XML Schema (XSD) or Tamino Schema 2 (TSD) as input Document Type.

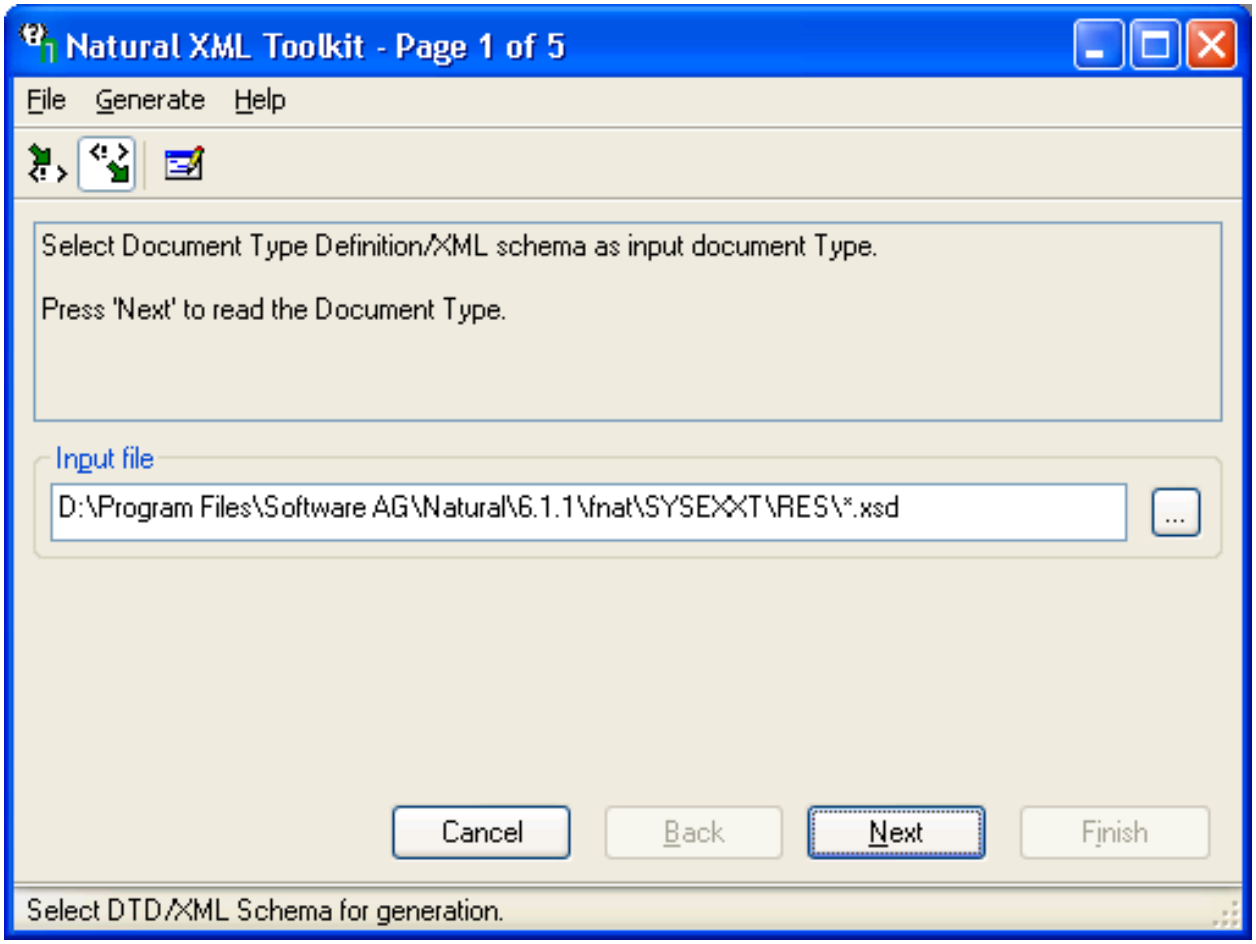
To invoke the dialog shown below

- Choose From DTD/XSD from the Generate menu.

  Or:

  Choose the button.

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



Field Descriptions

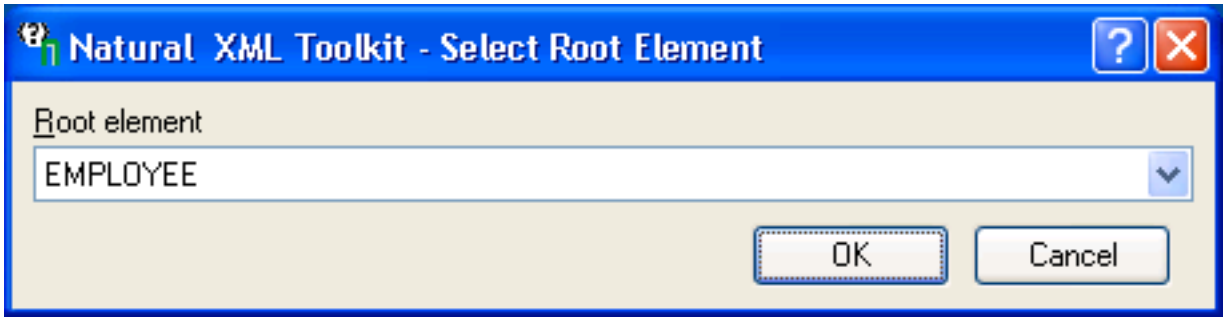
Input File

Select a DTD, XSD or TSD file. You can use the browse button ... to search for an existing DTD, XSD or TSD file.

Choose **Next** to continue.

Select Root Element or Document Type

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



Field Descriptions

Root Element (for DTDs)

Default Value:	(All Elements)

Select the desired element, e.g. EMPLOYEE, and choose **OK**.

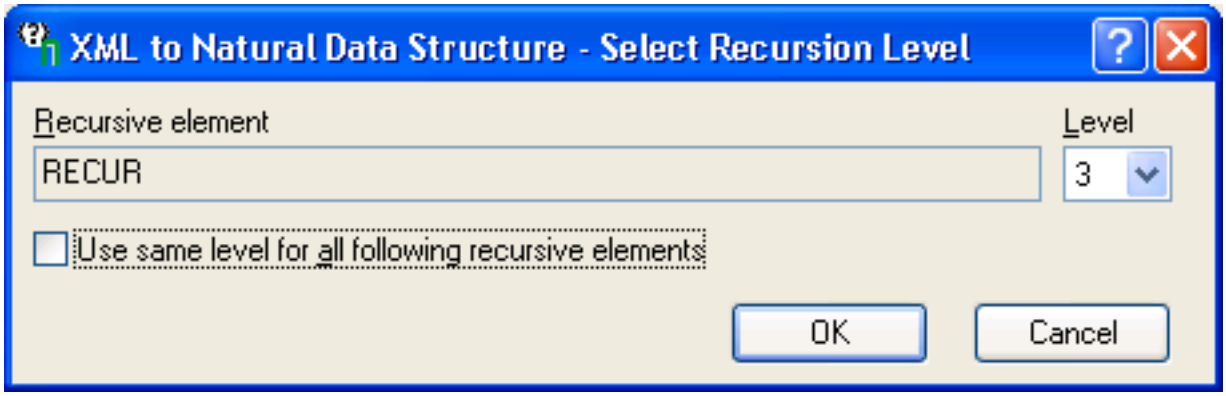
Document Type (for Tamino Schema)

Default Value:	(All Elements)

Select the desired element, e.g. EMPLOYEE, and choose **OK**.

Select Recursion Level

This dialog is only displayed, if the DTD, XSD or TSD selected in the first dialog includes recursive elements.



Field descriptions

Recursive Element

Name of the Element that is used recursively.

Default Value:	(All Libraries)

Level

Number of recursion levels that should be generated.

Default Value:	3

Use same level for all following recursive elements

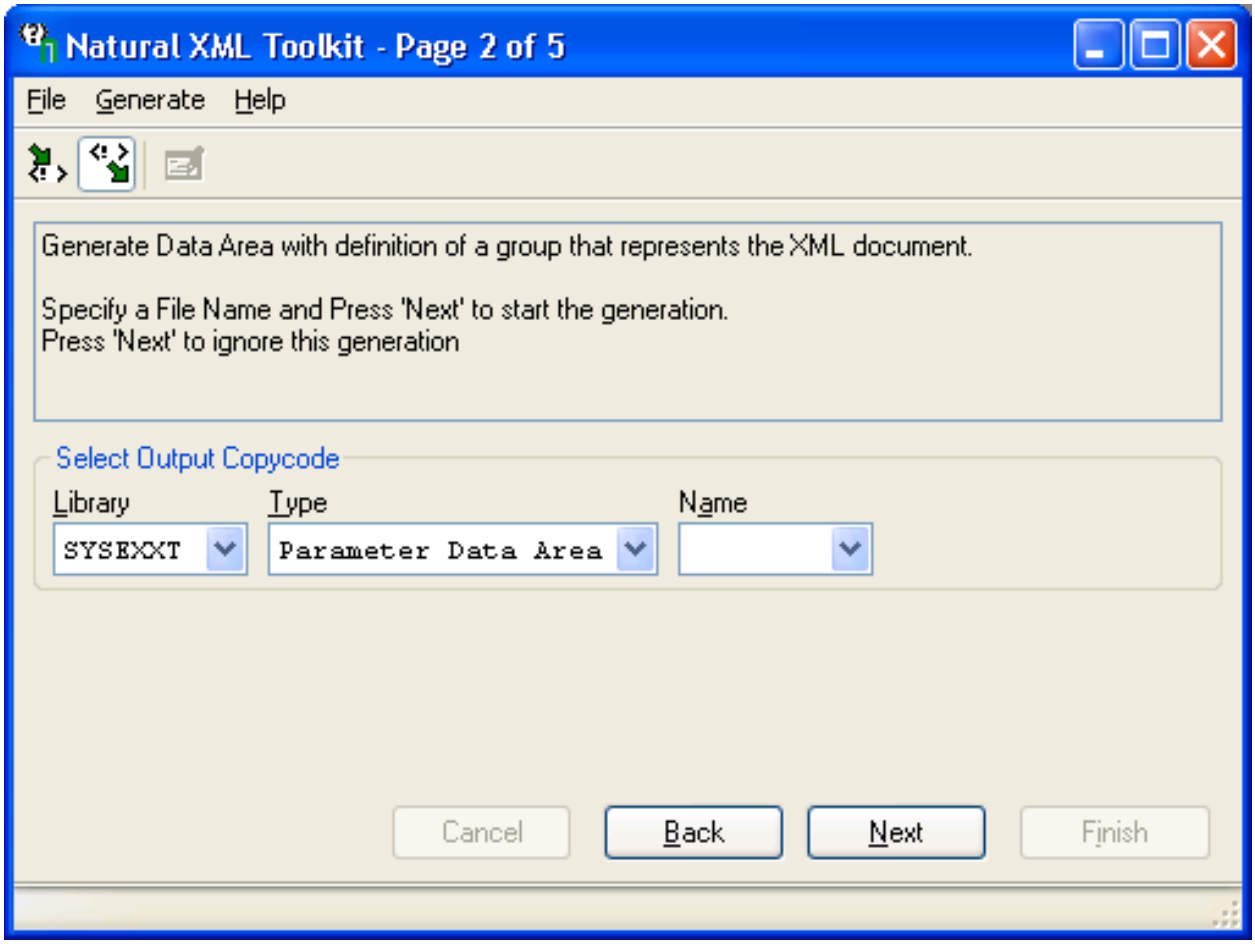
If another recursive element is found, the same recursion level will be used.

Default Value:	unchecked

Choose **OK** to continue.

Generate Natural Data Area

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



Field Descriptions

Library

Belongs to Group:	Select Output Copycode
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output Copycode
Default Value:	**L** - Local Data Area

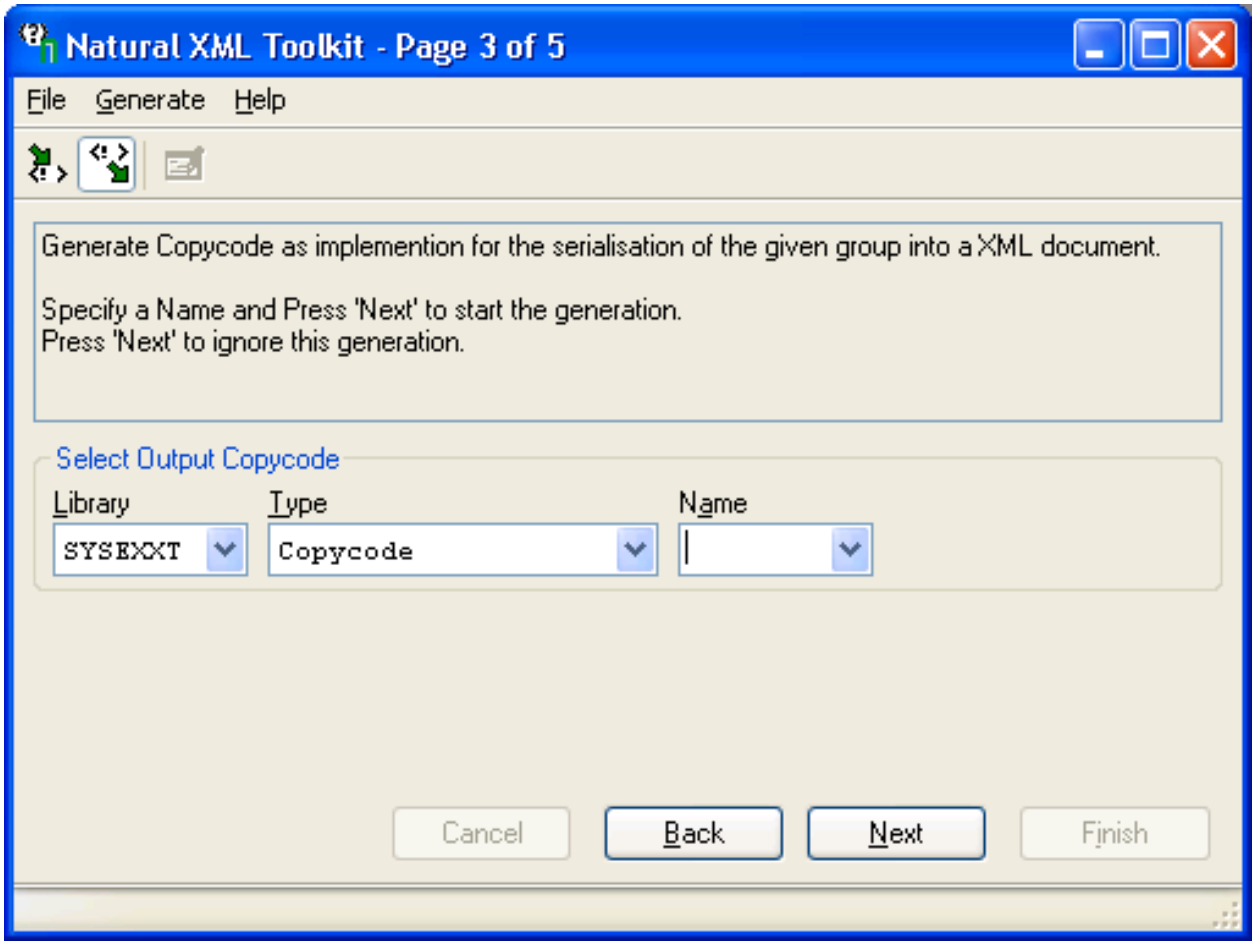
Name

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

Choose **Next** to continue.

Generate Copycode for Serialization

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



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

Field Descriptions

Library

Belongs to Group:	Select Output Copycode
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output Copycode
Default Value:	Copycode

Name

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

Choose **Next** to continue.

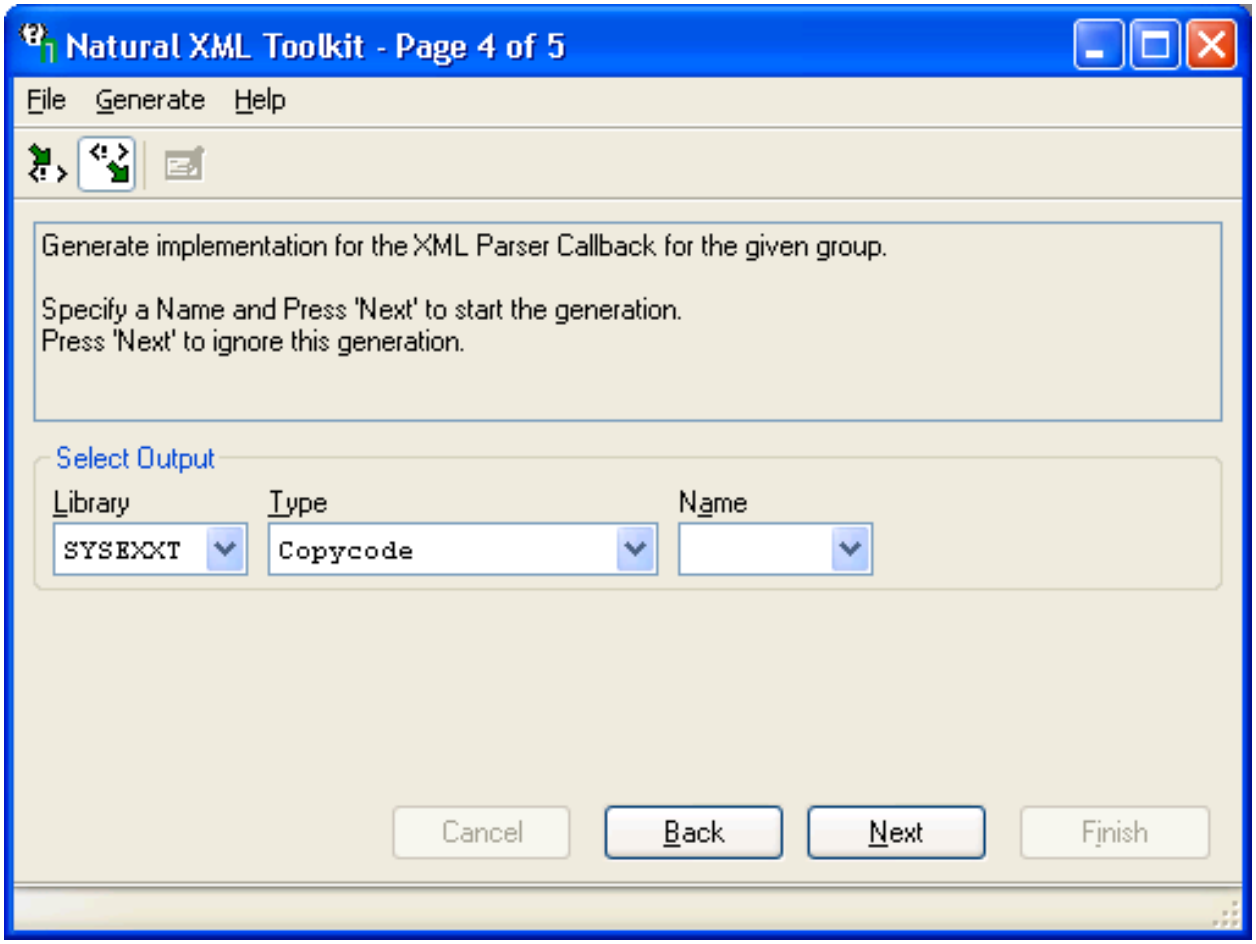
Generate Subprogram for Serialization

This dialog is used to generate a subprogram as implementation for the serialization of the given group into an XML document.

It uses the same entries as the above dialog for copycode, except that the Type field contains the entry Subprogram.

Generate Copycode for XML Parser Callback

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



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

Field Descriptions

Library

Belongs to Group:	Select Output
Default Value:	(All libraries)

Type

Belongs to Group:	Select Output
Default Value:	Copycode

Name

Belongs to Group:	Select Output
Default Value:	(All objects of the selected library and type)

Choose **Next** to continue.

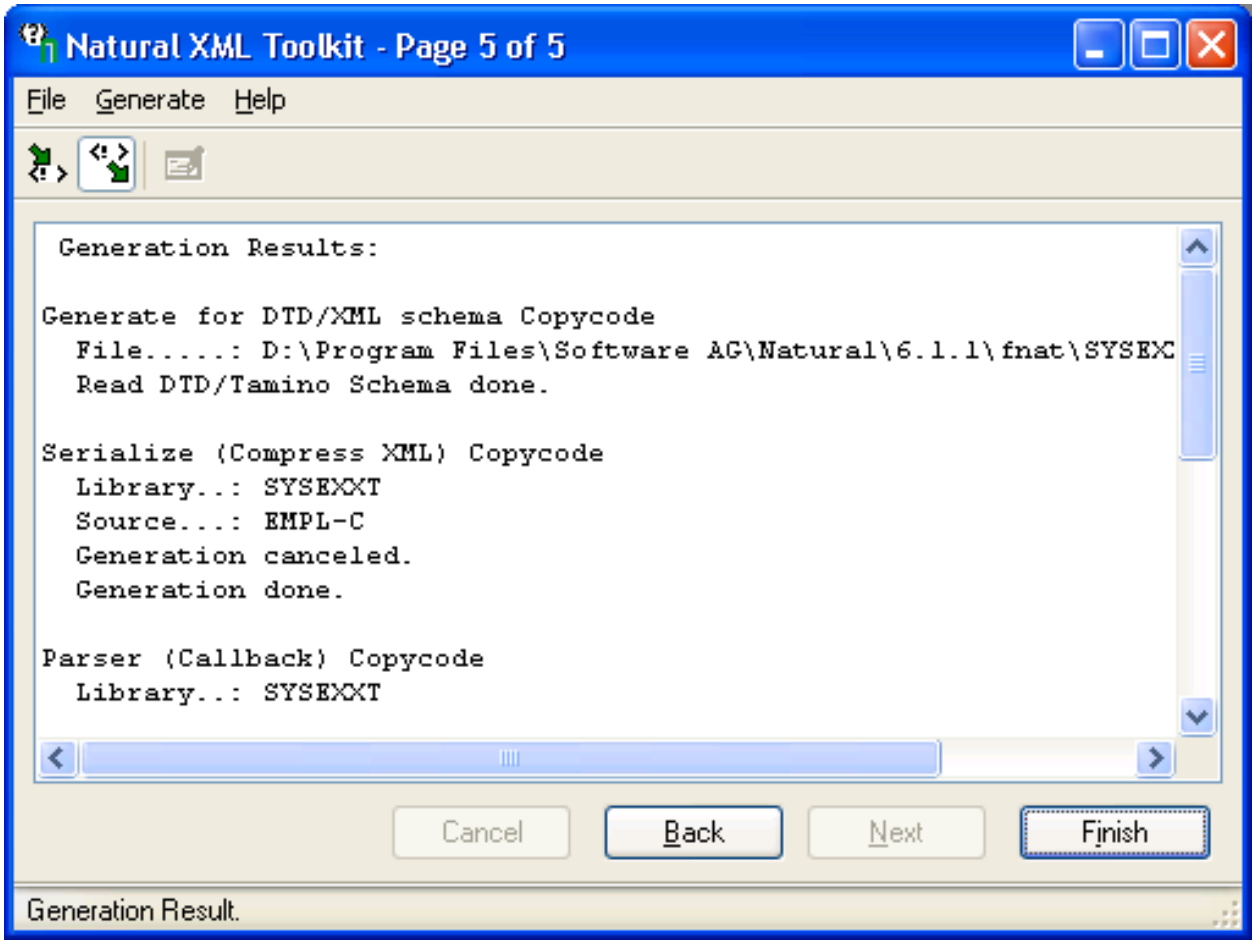
Generate Subprogram for XML Parser Callback

This dialog is used to generate a subprogram as implementation for the XML Parser Callback for the given group.

It uses the same entries as the above dialog for copycode, except that the Type field contains the entry Subprogram.

Show Generation Results

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



Choose **Finish** to end the generation process.

93 Natural Simple XML Parser

- Parser Description and Example 324
- Parser Restrictions 331

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
*
* (c) Copyright Software AG. 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)

94 Examples

- Serialize Copycode 334
- Serialize Subroutine 336
- Generated Natural Data Area 344
- Natural DTD Parser 346
- Generated Type Definition 347
- Parser CALLBACK Copycode 348

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
```

XML Schema Example:

```
<?xml
version="1.0" encoding="ISO-8859-1"?> <xs:schema ↩
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="EMPLOYEE"> <xs:complexType> <xs:sequence>
<xs:element ref="FULL-NAME"/> <xs:element ref="FULL-ADDRESS"/>
<xs:element ref="TELEPHONE"/> <xs:element ref="JOB-TITLE"/>
<xs:element ref="INCOME" minOccurs="0" maxOccurs="6"/>
</xs:sequence> <xs:attribute name="PERSONNEL-ID" use="optional">
<xs:simpleType> <xs:restriction base="xs:string"/> </xs:simpleType>
</xs:attribute> </xs:complexType> </xs:element> <xs:element
name="FULL-NAME"> <xs:complexType> <xs:sequence> <xs:element
ref="FIRST-NAME"/> <xs:element ref="NAME"/> </xs:sequence>
</xs:complexType> </xs:element> <xs:element name="FIRST-NAME">
<xs:simpleType> <xs:restriction base="xs:string"> <xs:maxLength
value="20"/> </xs:restriction> </xs:simpleType> </xs:element>
<xs:element name="NAME"> <xs:simpleType> <xs:restriction
base="xs:string"> <xs:maxLength value="20"/> </xs:restriction>
</xs:simpleType> </xs:element> <xs:element name="FULL-ADDRESS">
<xs:complexType> <xs:sequence> <xs:element ref="ADDRESS-LINE"
minOccurs="0" maxOccurs="6"/> <xs:element ref="CITY"/>
<xs:element ref="ZIP"/> <xs:element ref="COUNTRY"/>
</xs:sequence> </xs:complexType> </xs:element> <xs:element
name="ADDRESS-LINE"> <xs:simpleType> <xs:restriction base="xs:string">
<xs:maxLength value="20"/> </xs:restriction> </xs:simpleType>
</xs:element> <xs:element name="CITY"> <xs:simpleType>
<xs:restriction base="xs:string"> <xs:maxLength value="20"/>
</xs:restriction> </xs:simpleType> </xs:element> <xs:element
```

```
name="ZIP"> <xs:simpleType> <xs:restriction base="xs:string">
<xs:maxLength value="20"/> </xs:restriction> </xs:simpleType>
</xs:element> <xs:element name="COUNTRY"> <xs:simpleType>
<xs:restriction base="xs:string"> <xs:maxLength value="3"/>
</xs:restriction> </xs:simpleType> </xs:element> <xs:element
name="TELEPHONE"> <xs:complexType> <xs:sequence> <xs:element
ref="AREA-CODE"/> <xs:element ref="PHONE"/> </xs:sequence>
</xs:complexType> </xs:element> <xs:element name="AREA-CODE">
<xs:simpleType> <xs:restriction base="xs:string"> <xs:maxLength
value="6"/> </xs:restriction> </xs:simpleType> </xs:element>
<xs:element name="PHONE"> <xs:simpleType> <xs:restriction
base="xs:string"> <xs:maxLength value="15"/> </xs:restriction>
</xs:simpleType> </xs:element> <xs:element name="JOB-TITLE">
<xs:simpleType> <xs:restriction base="xs:string"> <xs:maxLength
value="25"/> </xs:restriction> </xs:simpleType> </xs:element>
<xs:element name="INCOME"> <xs:complexType> <xs:sequence>
<xs:element ref="SALARY"/> <xs:element ref="BONUS"
minOccurs="0" maxOccurs="4"/> </xs:sequence> </xs:complexType>
</xs:element> <xs:element name="SALARY"> <xs:simpleType>
<xs:restriction base="xs:string"> <xs:maxLength value="9"/>
</xs:restriction> </xs:simpleType> </xs:element> <xs:element
name="BONUS"> <xs:simpleType> <xs:restriction base="xs:string">
<xs:maxLength value="9"/> </xs:restriction> </xs:simpleType>
</xs:element> </xs:schema>
```

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
```

Serialize Subroutine

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

Subroutine Example EMPLP:

```
* ----------------------------------------------------------------------
* Generated from NATURAL XML TOOLKIT
*
*         'EMPLP'
*
* DESCRIPTION
*               XML Parser implementation
*               using PARSE XML statement for
*               datastructure 'EMPL'
*
* ----------------------------------------------------------------------
*
DEFINE DATA PARAMETER
1 #XML_INPUT             (A) DYNAMIC BY VALUE
PARAMETER USING EMPL
*
LOCAL
1 #XML_PATH             (A) DYNAMIC
1 #XML_VALUE            (A) DYNAMIC
*
LOCAL
1 #CX                         (I4)
1 #CY                         (I4)
1 #CZ                         (I4)
END-DEFINE
*
* ------------------------------------------------- INCLUDE THE PARSER
PARSE XML #XML_INPUT INTO PATH #XML_PATH VALUE #XML_VALUE
*
* DTD SYSEXXT EMPL
  DECIDE ON FIRST #XML_PATH
    VALUE 'EMPLOYEE'
      RESET EMPLOYEE
    VALUE 'EMPLOYEE/@PERSONNEL-ID'
      /* #IMPLIED
      EMPLOYEE.PERSONNEL-ID := #XML_VALUE
    VALUE 'EMPLOYEE/FULL-NAME'
      IGNORE
    VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME'
      IGNORE
    VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME/$'
      EMPLOYEE.FIRST-NAME := #XML_VALUE
    VALUE 'EMPLOYEE/FULL-NAME/NAME'
      IGNORE
    VALUE 'EMPLOYEE/FULL-NAME/NAME/$'
      EMPLOYEE.NAME := #XML_VALUE
    VALUE 'EMPLOYEE/FULL-ADDRESS'
      IGNORE
```

```
VALUE 'EMPLOYEE/FULL-ADDRESS/ADDRESS-LINE'
  /* optional multiple
  ADD 1 TO EMPLOYEE.C@ADDRESS-LINE
  EXPAND ARRAY EMPLOYEE.ADDRESS-LINE TO
    (1:EMPLOYEE.C@ADDRESS-LINE)
VALUE 'EMPLOYEE/FULL-ADDRESS/ADDRESS-LINE/$'
  #CX := EMPLOYEE.C@ADDRESS-LINE
  EMPLOYEE.ADDRESS-LINE(#CX) := #XML_VALUE
VALUE 'EMPLOYEE/FULL-ADDRESS/CITY'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/CITY/$'
  EMPLOYEE.CITY := #XML_VALUE
VALUE 'EMPLOYEE/FULL-ADDRESS/ZIP'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/ZIP/$'
  EMPLOYEE.ZIP := #XML_VALUE
VALUE 'EMPLOYEE/FULL-ADDRESS/COUNTRY'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/COUNTRY/$'
  EMPLOYEE.COUNTRY := #XML_VALUE
VALUE 'EMPLOYEE/TELEPHONE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/AREA-CODE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/AREA-CODE/$'
  EMPLOYEE.AREA-CODE := #XML_VALUE
VALUE 'EMPLOYEE/TELEPHONE/PHONE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/PHONE/$'
  EMPLOYEE.PHONE := #XML_VALUE
VALUE 'EMPLOYEE/JOB-TITLE'
  IGNORE
VALUE 'EMPLOYEE/JOB-TITLE/$'
  EMPLOYEE.JOB-TITLE := #XML_VALUE
VALUE 'EMPLOYEE/INCOME'
  /* optional multiple
  ADD 1 TO EMPLOYEE.C@INCOME
  EXPAND ARRAY EMPLOYEE.INCOME TO
    (1:EMPLOYEE.C@INCOME)
VALUE 'EMPLOYEE/INCOME/SALARY'
  IGNORE
VALUE 'EMPLOYEE/INCOME/SALARY/$'
  #CX := EMPLOYEE.C@INCOME
  EMPLOYEE.SALARY(#CX) := #XML_VALUE
VALUE 'EMPLOYEE/INCOME/BONUS'
  /* optional multiple
  #CX := EMPLOYEE.C@INCOME
  ADD 1 TO EMPLOYEE.C@BONUS(#CX)
  EXPAND ARRAY EMPLOYEE.BONUS TO
    (*:*,1:EMPLOYEE.C@BONUS(#CX))
VALUE 'EMPLOYEE/INCOME/BONUS/$'
  #CX := EMPLOYEE.C@INCOME
```

```
      #CY := EMPLOYEE.C@BONUS(#CX)
      EMPLOYEE.BONUS(#CX,#CY) := #XML_VALUE
    NONE
      IGNORE
  END-DECIDE
*
END-PARSE
*
END
```

Subroutine Example EMPL2S:

```
* ----------------------------------------------------------------------
* Generated from NATURAL XML TOOLKIT
*
*         'EMPL2S'
*
* DESCRIPTION
*               XML serialize implementation for
*               'EMPL' datastructure
*
* ----------------------------------------------------------------------
*
DEFINE DATA PARAMETER
1 #XML_SERIALZE_OUTPUT (A) DYNAMIC
PARAMETER USING EMPL
LOCAL
1 #CX                  (I4)
1 #CY                  (I4)
1 #CZ                  (I4)
END-DEFINE
*
#XML_SERIALZE_OUTPUT := '<?xml version="1.0" encoding="ISO-8859-1"?>'
*
* DTD SYSEXXT EMPL
COMPRESS #XML_SERIALZE_OUTPUT '<EMPLOYEE'
  ' PERSONNEL-ID="'EMPLOYEE.PERSONNEL-ID '"'
  '>' INTO #XML_SERIALZE_OUTPUT LEAVING NO
/* now the children
COMPRESS #XML_SERIALZE_OUTPUT '<FULL-NAME'
  '>' INTO #XML_SERIALZE_OUTPUT LEAVING NO
/* now the children
COMPRESS #XML_SERIALZE_OUTPUT '<FIRST-NAME'
  '>'
  EMPLOYEE.FIRST-NAME
  '</FIRST-NAME>' INTO #XML_SERIALZE_OUTPUT LEAVING NO
COMPRESS #XML_SERIALZE_OUTPUT '<NAME'
  '>'
  EMPLOYEE.NAME
  '</NAME>' INTO #XML_SERIALZE_OUTPUT LEAVING NO
/*
```

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

```
  FOR #CY = 1 TO EMPLOYEE.C@BONUS(#CX)
    COMPRESS #XML_SERIALZE_OUTPUT '<BONUS'
      '>'
      EMPLOYEE.BONUS(#CX,#CY)
      '</BONUS>' INTO #XML_SERIALZE_OUTPUT LEAVING NO
  END-FOR
  /*
  COMPRESS #XML_SERIALZE_OUTPUT '</INCOME>'
  INTO #XML_SERIALZE_OUTPUT LEAVING NO
END-FOR
/*
COMPRESS #XML_SERIALZE_OUTPUT '</EMPLOYEE>'
INTO #XML_SERIALZE_OUTPUT LEAVING NO
END
```

Program Example:

```
* ----------------------------------------------------------------------
* CLASS  NATURAL XML TOOLKIT
*
*
* DESCRIPTION
*               Serialize a given Data structure.
*
*
* AUTHOR        SAG
*
*
* (c) Copyright Software AG. 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:*)
    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:*)
    3 SALARY(A9)
    3 C@BONUS(I4)
    3 BONUS(A9/1:*)
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 Definition
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) 2 E1 (Aâ€¦/1:C@E1_MAX)	<!ELEMENT G1 (E1*)> <!ELEMENT E1 (#PCDATA)>
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€¦)	<!ELEMENT E1 (#PCDATA)> <!ATTLIST E1 A1 #FIXED "schema" A2 NMTOKEN #IMPLIED A3 ID #REQUIRED>

Natural	Document Type Definition
4 A3 (Aâ€¦)	

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 dialog** 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 **er**.

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
*
*
* (c) Copyright Software AG. 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
```

95 XML 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