Conversion Program HTML to Natural with SYSWEB3

Conversion Program HTML to Natural with SYSWEB3

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

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

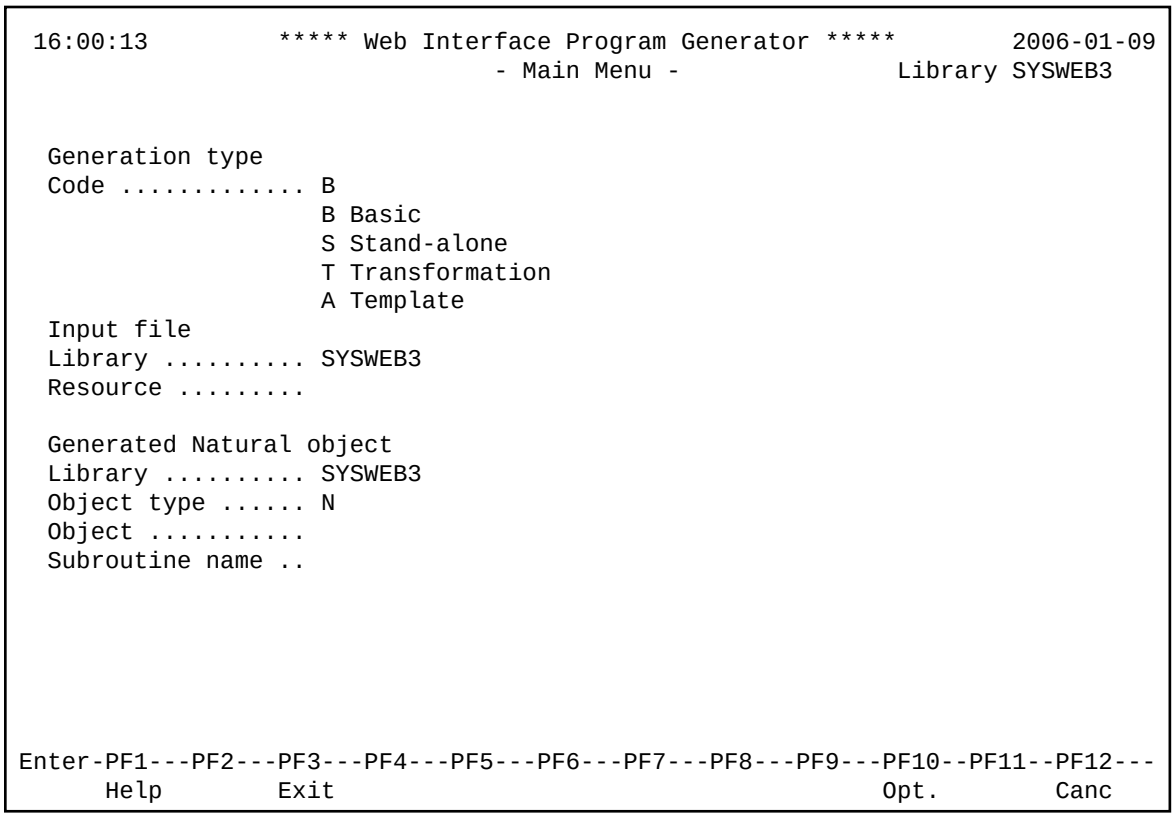
This section covers the following topics:

- Using the Conversion Program
- Inserting a Natural Tag
- Inserting Replacement Strings
- Options
- Online Test Utility WEB-ONL3

Using the Conversion Program

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

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



Below is information on:

- Functions and Keys
- Generating a Subprogram/Subroutine to be called directly from the Web
- Generating a Standalone Subprogram to be Called Directly from the Web
- Generating a Subprogram/Subroutine using Natural Tags

Functions and Keys

Key	Function	Explanation
PF1	Help	Invokes the Help function for the field at which the cursor is positioned.
PF3	Exit	Leaves the program and returns to the command line.
PF10	Opt.	Options. Specifies options for the generation process.
PF12	Canc	Leaves the program without saving your changes.
ENTER		Starts generating the program.

Generating a Subprogram/Subroutine 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. 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: Stand-alone.

2. Select your Generated Natural object.
3. Start the generation.
4. 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. Choose the Natural library you want to generate to.
4. Select the object type you want to generate.
5. Select your Generated Natural object.
6. Start the generation.
7. After the generation, you can call the Natural Web Interface to show the page.

Inserting a Natural Tag

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

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

Appearance

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

Below is information on:

- Attributes DATA, LDA, GDA, SUB, NOT
- Comment Tag
- ASP-like Script Commands
- Additional Script Directives
- Example 1 of a Simple Generation
- Example 2 of a Simple Generation with a Natural Tag
- Generating a Subprogram/Subroutine using a Template that is Called Directly from the Web

Attributes DATA, LDA, GDA, SUB, NOT

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

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

Comment Tag

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

ASP-like Script Commands

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

Additional Script Directives

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

Output directive: <%= ... %>

Short form for <% PERFORM W3HTML ... %> tag

Subprogram directive: <%SUB ... %>

equal to the <NATURAL SUB> ... </NATURAL> tag

Global Data Area directive: <%GDA ... %>

equal to the <NATURAL GDA> ... </NATURAL> tag

directive: <%LDA ... %>

equal to the <NATURAL LDA> ... </NATURAL> tag

Not directive: <%NOT ... %>

equal to the <NATURAL NOT> ... </NATURAL> tag

Processing directive <%@ LANGUAGE=NATURAL %>

indicates that the used language is Natural.

Example 1 of a Simple Generation

HTML document:

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

Generated Natural subprogram:

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

Example 2 of a Simple Generation with a Natural Tag

HTML document:

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

Generated Natural subprogram:

```
* ----- GENERATED BY NATURAL WEB INTERFACE
* File ..........: E:\SAG\Natural\6.2\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. Select the object type you want to generate.
4. Select your Generated Natural object.
5. Start the generation.
6. After generation, you can call the Natural Web Interface to show the page.

Inserting Replacement Strings

It is neccesary 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\6.2\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\6.2\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

```
 15:56:29                ***** HTML to Natural *****               2006-01-09
 User DEFAULT                    - Options -                Library SYSWEB3

  HTML File
  Delete unnecessary white space ..............
  Save <NATURAL NOT>..</NATURAL> at source .....

  Generated Source
  Stow after generation ....................... X
  Use long constants .......................... X
  Natural line length ......................... 128


  Template
  Delimiter ................................... $


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

Below is information on:

- Input/Output Fields
- Functions and Keys

Input/Output Fields

Field	Explanation
Delete unnecessary white space	If checked, multiple white-space characters such as blank, new line, tab, will be reduced to a single white space. For special HTML tags such as <PRE>, <TEXTAREA> or <SCRIPT>, the white space will not be collapsed. Default value: unchecked
Save <NATURAL NOT> ... <NATURAL> in Source	If checked, the content of <NATURAL NOT> tags will not usually be generated into the Natural source. This option generates the content of <NATURAL NOT> as comment into the Natural source. Default value: unchecked
Stow after Generation	If checked, the generated program will be stowed if the generation has been successful. Default value: checked
Natural Line Length	The length of the generated Natural source lines: the minimum value is 20, the maximum 246. Default value: 72
Default Input File	The default input file to be used for the generation. Default value: /nat/natc/611/samples/sysexxt

Functions and Keys

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

Online Test Utility WEB-ONL3

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

Features:

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

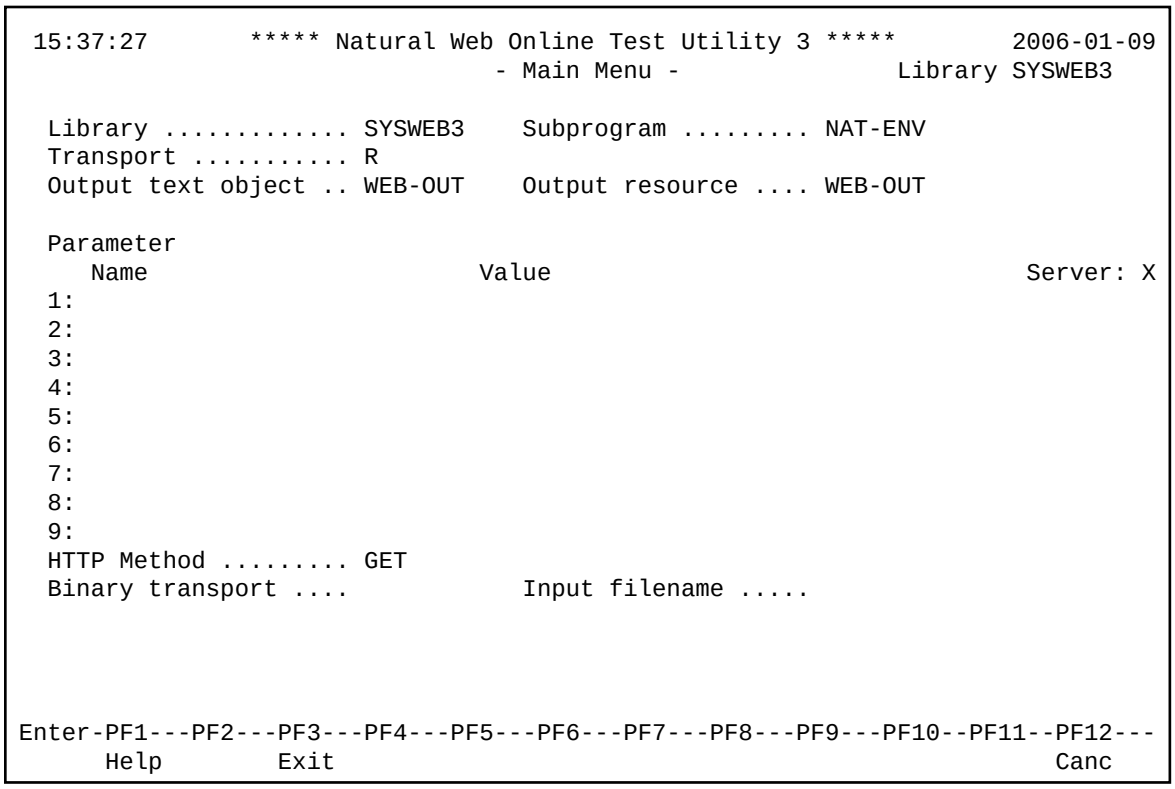
Below is information on:

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

Running the Application

To run the application

1. Start the program WEB-ONL3.
2. Select a library and subprogram name.
3. Optional: add parameters.
4. Choose RPC.
5. Press ENTER.



Input/Output Fields

Field	Explanation
Library	The library or class in which the required subprogram is stored.
Subprogram	The name of the required subprogram or method.
Transport	Uses RPC as communication form. Value R is preselected.
Output text object	The name of the Natural object of the type Text that stores the result of the generated web page. Default: WEB-OUT
Output resource	The name of the Natural object of the type Resource that stores the result of the generated web page. Default: WEB-OUT
Parameter: Name Value Server	Here you can enter the name-value-pairs needed from the subprogram. If you use server parameters, place an ampersand (&) in front of the variable name before you add the parameter to the parameter list. Server: If any of the name-value-pairs are server variables, you need to check option. Note that any status will last until you change it again.

Field	Explanation
HTTP Method	Here 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). Use the binary transport option, 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. A mime type which has been set manually will always override the default mime type. **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 input file must be specified. Use the binary transport option, if you want to submit binary data.

Functions and Keys

Key	Function	Explanation
ENTER		Runs the process of receiving the output of the requested subprogram. The status of the process can be seen in the message line at the bottom of the WEB-ONL3 program screen.
PF3	Exit	Leaves the Test Utility and returns to the command line.
PF12	Canc	Cancel. Stops processing.