FUJITSU

FUJITSU Software NetCOBOL V11.0

Web Guide

Windows

B1WD-3363-01ENZ0(00)
August 2015

Preface

This manual contains an overview of Web applications using COBOL and the features of Web Servers.

For details regarding the specifications of NetCOBOL Web subroutines, refer to the *NetCOBOL CGI Subroutines User's Guide*, and *NetCOBOL ISAPI Subroutines User's Guide*.

Supported Operating Systems

The NetCOBOL Web link functions provide the environment for developing and executing applications on Web Servers.

Registered Trademarks

The trademarks used in this manual are listed as follows:

- NetCOBOL is a trademark or registered trademark of Fujitsu Limited or its subsidiaries in the United States or other countries or in both.
- Microsoft, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Other company names and product names are the trademarks and registered trademarks of the companies.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. Oracle Solaris might be described as Solaris, Solaris Operating System, or Solaris OS.
- Microsoft Corporation has permitted Fujitsu Limited to use the screens.

Product Names

The product names used in this manual are abbreviated as follows:

Product Name	Abbreviation
Microsoft® Windows Server® 2012 R2 Datacenter Microsoft® Windows Server® 2012 R2 Standard Microsoft® Windows Server® 2012 R2 Essentials Microsoft® Windows Server® 2012 R2 Foundation	Windows Server 2012 R2
Microsoft® Windows Server® 2012 Datacenter Microsoft® Windows Server® 2012 Standard Microsoft® Windows Server® 2012 Essentials Microsoft® Windows Server® 2012 Foundation	Windows Server 2012
Microsoft® Windows Server® 2008 R2 Foundation Microsoft® Windows Server® 2008 R2 Standard Microsoft® Windows Server® 2008 R2 Enterprise Microsoft® Windows Server® 2008 R2 Datacenter	Windows Server 2008 R2
Microsoft(R) Internet Information Services	IIS
Microsoft(R) Internet Explorer	IE

Export Regulation

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Contents

Chapter 1 Overview

Today's business applications are appearing more and more on the Internet.

The system configuration where these Web applications are typically executed includes host-centralized systems (Web Servers), client systems, and network-computing system support.

The network-computing system support has a role of providing access to information as a result of the rapid increase of host computers connected to the Internet. The network-computing environment enables businesses to be mutually linked through a network, and business efficiency to be improved as a result of the shared information involving the Web. Web Browsers with platform-free operability are essentially used in mutually linking various types of businesses.

For host-centralized systems and client/server systems, COBOL is commonly used as the core language in the mission critical area of business systems.

On the basis of Internet/Intranet technologies, the demand on COBOL to support the business enterprise has been increasing.

1.1 Effectively Using the Internet/Intranet

To effectively use Internet/Intranet environments, the features of the Web and the merits of using COBOL are explained as follows.

1.1.1 Features of the Web

Remote site

Using Web Browsers, all resources on the network can be accessed, so that client system can be easily connected to remote systems.

Cost reduction

Using Web Browsers, client machine setup/maintenance work can be simplified.

Independence from the client system

Using Web Browsers, a business system independent of the client system can be built. These systems can also be used in the mobile computing environment.

Distribution of the load on the server

The server, if it can be connected to the network, enables various business systems to be distributed.

Security

As one of the features, the Web can be easily connected to the client system. In this case, however, there is a problem regarding security. When building such a system, self-defense measures need be taken, for example, limiting the access to the domain to which the client is connected via Web Servers, or setting user restrictions in the application.

1.1.2 Merits of using COBOL

Generally, when selecting the language for developing applications to execute in Web environments, the following features are considered to enable the data input/output with Web Browsers:

- Data manipulation
- Access to files or databases
- Environment variables manipulation

COBOL satisfies the above features and can be used to develop the above type of applications efficiently, considering the following facts:

Use of accumulated knowledge or know-how for building a system

As the know-how previously accumulated while developing an application by using COBOL can be used, the development process can be sped up.

Effective use of existing resources

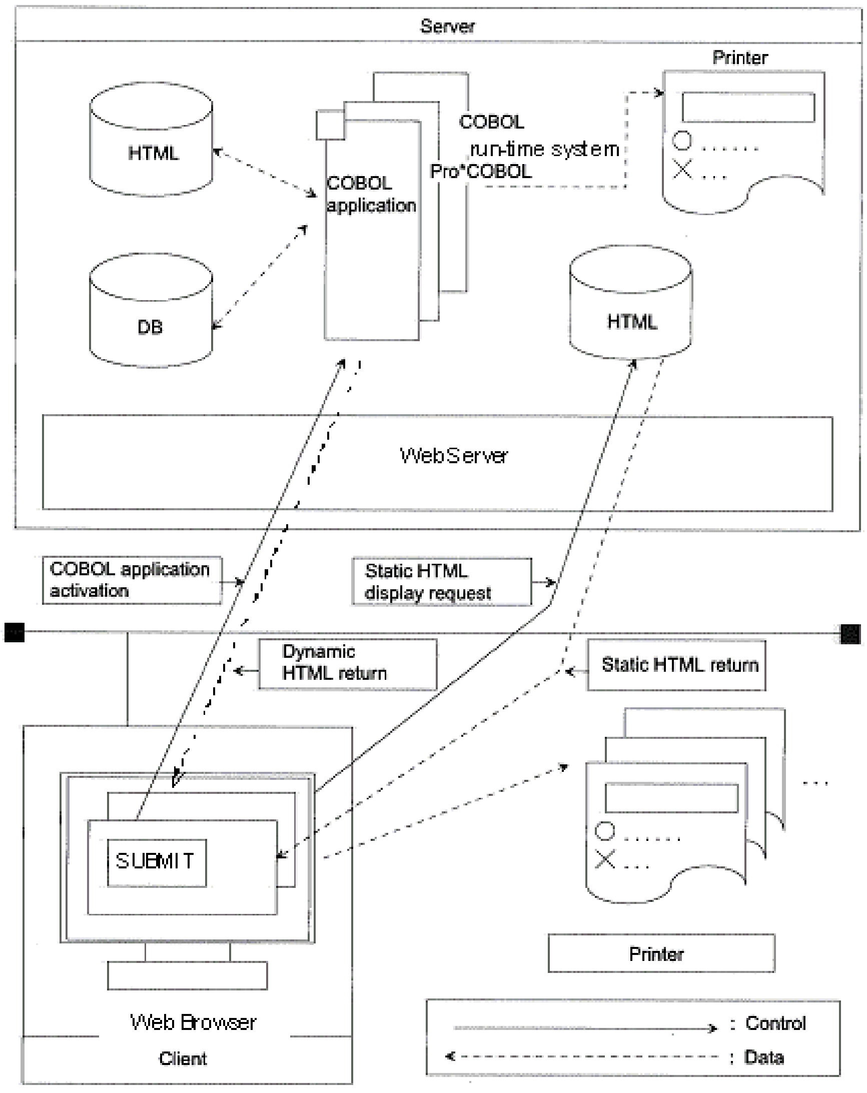
The resources used by the business system in the conventional environment can be used effectively.

1.2 Web Applications

HTML documents can return only static information, so the updated information cannot be communicated in real-time. On the contrary, information can be provided in real-time by using CGI and APIs peculiar to various Web Servers for the applications called from Web Browsers. These applications can dynamically return updated information related to the information input from Web Browsers. In addition to the information returned, the data processing procedure including the file input/output and database access, which is essential to the business application, can be requested (or executed) from Web Browsers.

COBOL is the business processing language used mainly for data processing, and dynamic Web pages can be created interactively by effectively using these features of COBOL.

Web applications are executed under the services of Web Servers. For the notes on creating or executing these types of applications, see Net*COBOL User's Guide*.



1.3 How to Build Applications for the Web

Use the following two methods for building a business system in COBOL for execution on the Internet/Intranet.

Development of the application using the COBOL Web subroutines

Using this method, applications can be generated by using COBOL knowledge only, irrespective of complicated processing. For example, the processing and of parameters required for developing the application by using data input/output with Web Browsers, and outputting of the processing results.

Chapter 2 Applications Using Web Subroutines

For development/maintenance of Web applications, knowledge regarding interfaces with Web Browsers is generally required. Using NetCOBOL Web subroutines, data can be input/output between the COBOL programs and Web Browsers by using COBOL knowledge only.

2.1 Interface Types Available by Web Applications

There are many different interfaces available with applications on Web Servers, as explained below:

ISAPI (Internet Server API)

ISAPI is an extended programming interface for interactive applications executing on Microsoft's Internet Information Server (stated as IIS hereafter) Web Server, and was first proposed by Microsoft. NetCOBOL provides ISAPI and its execution codes, which are encapsulated according to the COBOL language specification. ISAPI subroutines provide functions for more easily developing ISAPI applications to be executed on IIS.

ISAPI resembles CGI in that CGI is a programming interface for developing interactive applications to be executed on Web Servers, but differs from CGI in overall resource usage, including CPU and memory as well as performance. For details of the differences between ISAPI and CGI, see "Differences between ISAPI and CGI" in this manual.

CGI (Common Gateway Interface)

CGI is the most commonly used interface for data input/output with Web Servers.

Most Web Servers support CGI.

NetCOBOL provides COBOL CGI subroutines as functions for more easily developing applications using CGI.

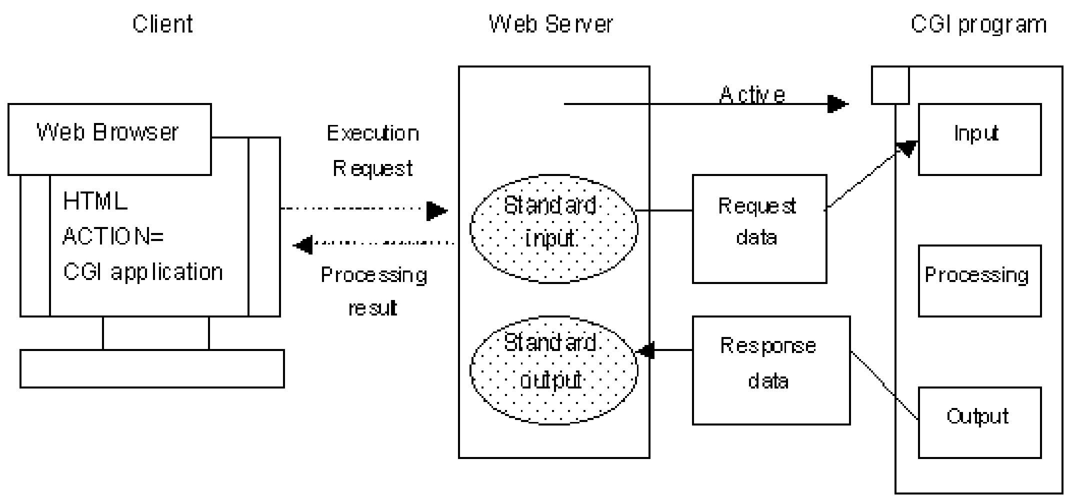
2.2 Processing Flow of Web Applications

The following explains Web Browsers and the process flow of Web applications, using CGI as the example.

CGI executes on the Web Server and is used for data input/output with Web Browsers. CGI is the most fundamental interface for executing CGI programs (called CGI scripts) on Web Servers from Web Browsers (clients). A request to the Web Server from a Web Browser provides the function of activating the CGI program and inputting/outputting the data (including the data input via environment variables).

CGI is executed in the processing flow of inputting the data (request), processing the data, and outputting the processing result (response), and the CGI program must be cognizant of this processing flow.

The processing flow by ISAPI and other APIs is basically the same as that of processing by CGI. However, CGI inputs/outputs the data via the standard input-output, and ISAPI or other APIs provide a special API (hidden in API) used for inputting/outputting the data. As this difference in data inputting/outputting between CGI and ISAPI or other APIs has been absorbed by the NetCOBOL Web subroutines, COBOL applications need not recognize this difference if the NetCOBOL Web subroutines are used.

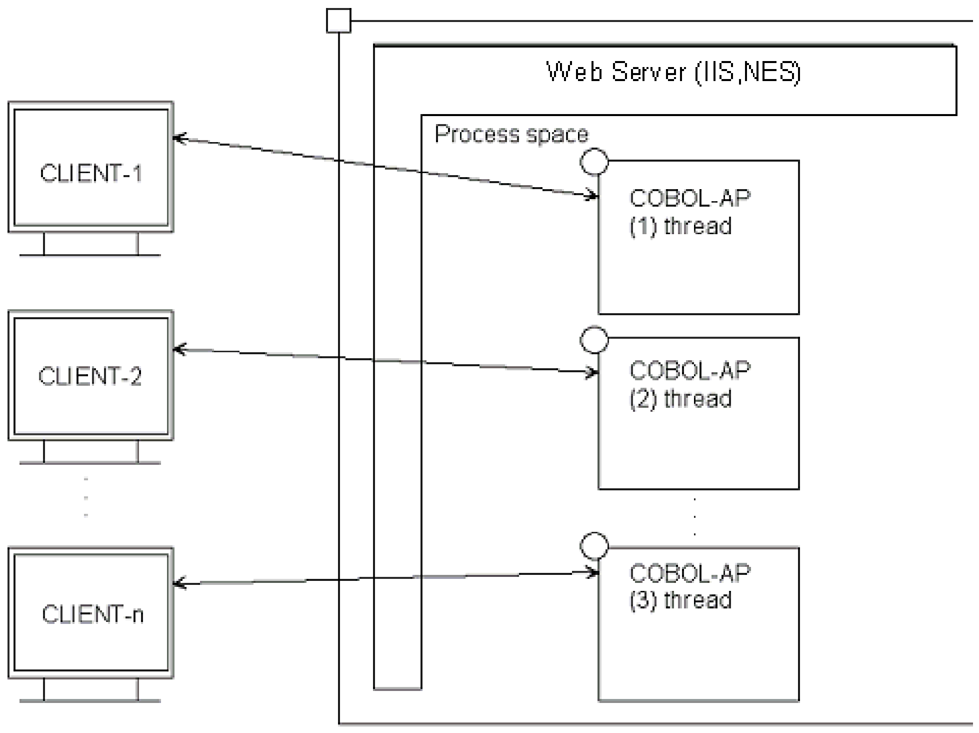


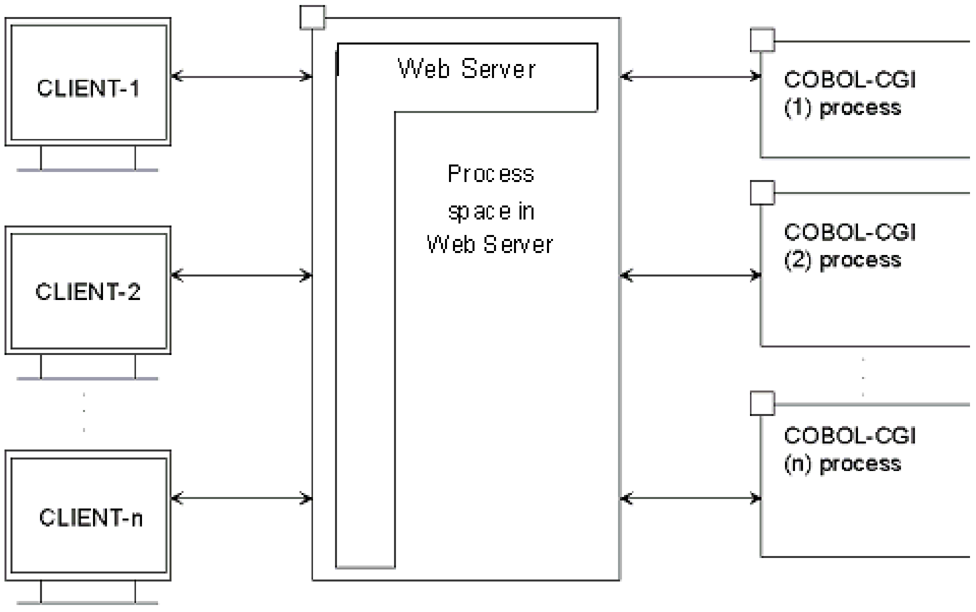
2.3 Differences between ISAPI and CGI

CGI applications must have an executable module format (.EXE). Therefore, the Web Server generates a separate execution process for each CGI application for each request from the client (Web Browser), and as a result, resources including CPU and memory in the server are consumed in large quantities.

In contrast, ISAPI applications are generated using dynamic link library format (.DLL) and loaded into one overall process in the Web Server. Then, the loaded DLL(s) is controlled by being loaded directly into memory, or being unloaded after a lapse of specified time.

The major difference between ISAPI and CGI is that CGI is executed in separate processes for each request, but ISAPI generates a thread within a common address space in Web Servers and is executed in a multithreaded environment . As a result, the overhead caused by process generation/deletion in CGI, memory allocation, and task switching can be greatly reduced, and Web applications with higher performance can be built.





2.4 Purpose of Using the NetCOBOL Web Subroutines

The NetCOBOL Web subroutines provide functions for analyzing Web parameters that have been input from Web Browsers and are required for developing Web applications. These also include functions for outputting the processing result. Using these functions, Web applications can be generated easily by using only COBOL, without requiring specialized knowledge.

2.5 NetCOBOL Web Subroutines

2.5.1 Use of ISAPI

Using the NetCOBOL ISAPI subroutines, Web applications using ISAPI to execute on IIS can be generated.

NetCOBOL ISAPI subroutines are executed via the interface common to the NetCOBOL CGI subroutines, so the conversion from CGI to ISAPI can be easily performed (where CGI specification is almost same as ISAPI specification, but some changes specific to ISAPI need be added).

2.5.2 Use of CGI

Using the NetCOBOL CGI subroutines, Web applications using CGI executing on various Web Servers can be generated.

Chapter 3 Points of Selecting the COBOL Web Link Function

3.1 Features of Various Web Link Functions

The following table lists the features of COBOL Web link functions.

		Link function		
		ISAPI subroutines		CGI subroutines
Features	Web Server that can be used	IIS		All servers supporting CGI
	Web Interface technology	ISAPI		CGI
	File format activated by Web Server	DLL		EXE
	Operation model	Thread		Process
	Loading mechanism of COBOL applications executed on Web Servers	It is loaded at the first request to the application, then resides in memory for the next request.		It activates a process for each request to the application, and discards the process after processing ends.
	Execution performance	High-speed performance by multithreaded operation or DLL pre-load function.		Low-speed performance due to the overhead caused by a separate process activated/deleted for each request.
	User/Application Interface	- HTML - Web subroutines		
	Ease of generating a business application	Because the interfaces with Web Servers are hidden in Web subroutines complying with the COBOL language specification, no special knowledge regarding Web is required.		
	Utilization of existing resources	The actual business logic of existing applications can be mostly used without being modified significantly. However, interfaces with Web Servers are required, so the description for using Web subroutines must be added and the business logic must be organized to make use of such.		
	Forms printing	The server can execute forms printing contained in the application. However, the printing function of the Web Browser or the downloaded application may be used by the client.		
	Merit	Applications with high performance and resources consumed in small quantities can be generated by multithreaded operation.		Applications with high independence can be generated, independent of the server.

Note

The above table shows the outlined information for arranging the features of various Web link functions in rows.

For more detailed information, related products, and notes, refer to the manual corresponding to each Web link function.

3.2 Detailed Information for Each Web Link Function

If you decide that a COBOL Web link function should be used, refer to the manual in which the detailed information for each link function and design/develop your business application as documented.

The following lists the manuals corresponding to the Web link functions.

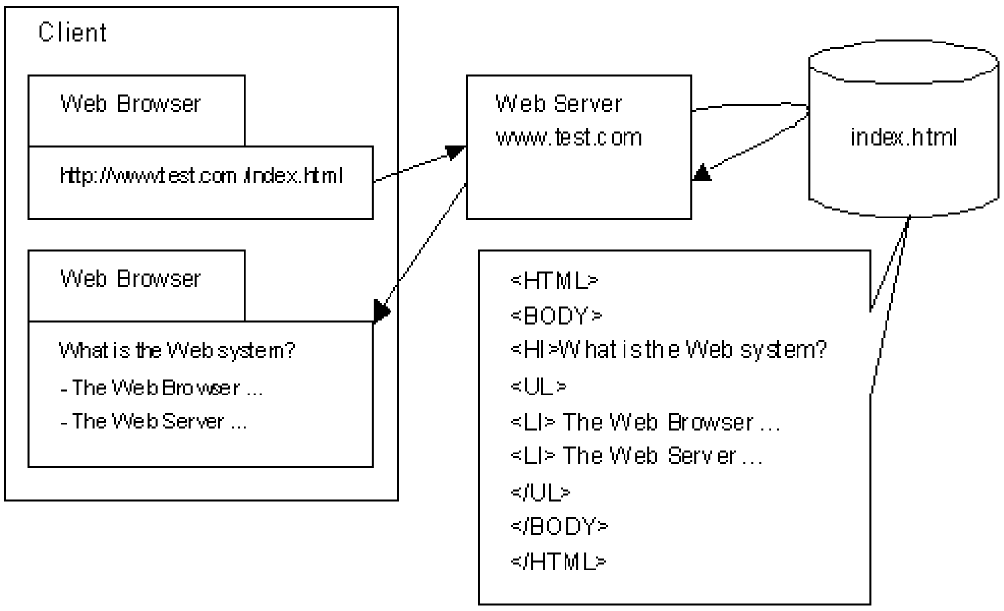
- When COBOL ISAPI subroutines are to be used, refer to the *NetCOBOL ISAPI Subroutines User's Guide* for details of the COBOL ISAPI subroutines.
- When COBOL CGI subroutines are to be used, refer to the *NetCOBOL CGI Subroutines User's Guide* for details of the COBOL CGI subroutines.
- When an application developed by using NetCOBOL CGI Subroutines is to be converted to an application using COBOL ISAPI subroutines, refer to Appendix D. CGI To ISAPI Subroutines Conversion Guide in the NetCOBOL ISAPI Subroutines User's Guide for the conversion from CGI to ISAPI

Appendix A For Beginners Developing COBOL Web Applications

This appendix explains fundamental matters for beginners who attempt to build COBOL applications using the Web, and for beginners who wish to use the COBOL Web subroutines.

A.1 What is a Web Application?

The WWW used by the Internet/Intranet connects Web servers on networks to Web Browsers (clients) by using HTTP (Hyper Text Transfer Protocol). Web Browsers enable access to a specific resource on a specific Web Server by specifying URL's (Uniform Resource Locators). Web Servers send the requested resource (typically an HTML page) back to the requesting Web Browser. Normally, Web Servers send HTML (Hyper Text Markup Language) data, so that Web Browsers can display the contents of the HTML data in an appropriate format.



In this way, the use of Web Browsers enables the information published by the connectable server to be accessed. At present, Web Browsers enable the information to be freely accessed, and can be used as the environment for executing a business application.

When only the HTML document is put on the server, static information can be opened to the public. However, it cannot be processed to access to the data base on an indispensable server for the business and to generate the HTML document dynamically only by static information. The main interface of Web is shown below.

- CGI (Common Gateway Interface)
- ISAPI (Internet Server Application Programming Interface)

In the business application, it is necessary to make the application in Web Server by using these interfaces. The application executed on this Web Server is called Web application.

A.2 NetCOBOL Web Subroutines

NetCOBOL Web subroutines provide the interfaces for generating Web applications using COBOL and the environment for executing these Web applications. NetCOBOL Web subroutines consist of various interfaces including CGI and ISAPI. Being able to use these interfaces without being conscious of the differences between CGI and ISAPI, however, allows the user to develop Web applications more easily than using other languages such as Perl and C.

NetCOBOL Web subroutines have the following features:

- Being able to be executed by the CALL statement interface, NetCOBOL Web subroutines can be used easily by COBOL.

- Using the functions provided for converting parts of HTML documents and for outputting the HTML document, the program and HTML document can be made independent, and development with high productivity and maintainability can be realized.
- The user can generate Web applications operating on various Web servers by describing a program with the interface provided by the NetCOBOL Web subroutines.

NetCOBOL Web subroutines provide the following services:

- Receiving the request data (Web parameters) passed from a Web Server when activating the application, and converting this data into the proper COBOL data format.
- Outputting the processing result (e.g. dynamic HTML output function)
- Automatically generating the header (e.g. Content-Type)
- System command execution function
- Obtaining request information
- Receiving/Outputting Cookie data
- Uploading files

A.3 HTTP Basics

The Web is established through the use of HTTP. The data transmission between Web Servers and Web Browsers is performed through HTTP. The data send from a Web Browser to a Web Server is called a "request," and the data sent from a Web Server to a Web Browser is called a "response." Generally, a response must exist for each request. A communication path is assured by the request, and released when a response is completed. For HTTP, the request and response have the same format except the first line, which have very simple formats as shown below.

Request Data

Method URL version
HTTP header
Message body

Response Data

Version Status code Reason
HTTP header
Message body

The following explains each of the terms used in the above tables:

Method

Indicates the contents of a request to a Web Server, which include GET, POST, and HEAD.

URL

Indicates the location of a resource to be requested.

Version

Indicates the version of HTTP to be used.

Status code

Indicates the condition of processing by the server or Web application, which is represented in 3 digits.

Reason

The reason for the status code.

HTTP header

Used to control Web Browser from Web Server, or to add the conditions to the Web Browser information or method.

Message body

Indicates the data to be transmitted between a Web Server and a Web Browser.

For details of these terms, refer to the specifications or books about HTTP. This appendix briefly explains the method, status code, and HTTP header that are referred to or set up by Web applications.

The following lists the main methods:

GET

This method fetches the information specified in URL. When the information specified in URL is a Web application (an executable file name and optional input parameters), this method resends the data output by the Web application.

There is an upper limit to the amount of data that the Web server can receive through the data input to the form that is passed to the application.

HEAD

This method is functionally the same as GET, but different from GET in the respect that only the HTTP header is resent. This method is used for checking the attribute of the information specified in a URL.

POST

This method is used for sending the data input from forms, to the Web application. GET functions similar to POST, but the quantity of data transmitted by GET is upper-limited.

Table A.1 Status Codes

Status code	Explanation
200	Normal end
302	The requested document was temporarily moved. For the location where it was moved, refer to the "Location" header.
303	The requested document was moved to a different URL. For the location where it was moved, refer to the "Location" header.
400	Syntax error
401	Authorization has failed.
403	Accessing the specified URL is inhibited.
404	There is no resource for the requested URL.
410	There is a resource inconsistent between the Web Browser and Server.
500	Internal error. For example, when no response is sent from Web applications.
502	An invalid request was returned from the Web Server.
503	Web Server cannot be accessed because of high load.

Table A.2 HTTP Headers

HTTP header	Explanation
Accept	Indicates the MIME type that Web Browser can received.
Accept-Charset	Indicates the character code set that Web Browser can received.
Accept-Encoding	Encoding format that Web Browser can received, which generally indicates the data compression format.
Accept-Language	Indicates the language type that Web Browser can received. The national language is represented as "ja" and English is represented as "en."

HTTP header	Explanation
Content-Encoding	Indicates the encoding format of the response data.
Content-Language	Indicates the language type of the response data.
Content-Type	Indicates the MIME type of the response data.
Date	Indicates the date when the request data or response data has been created.
Expires	Indicates the term of validity for the response data.
Location	Indicates the correct location where the information is stored.
Host	Host name of the server where Web Server is operated.
Referer	Indicates the requesting URL.
User-Agent	Indicates the Web Browser information.

A.4 HTML Basics

HTML (Hyper Text Markup Language) is the language for representing hyper text, and the text style or link is represented with a specific keyword called "tag." A tag is represented with a character string placed between "<" and ">." Also, this character string is called "tag-name."

In most cases, the tag has a start and end tag, which are represented as "<tag-name>" and "</tag-name>," respectively. The character string placed between the start and end tag is significant for Web Browsers. There is a tag called "single tag," which is significant by merely representing as "<tag-name>." In addition, the tag may have one or more attributes. An attribute is mostly represented in the format of "attribute-name=attribute-value," but it may be represented in the format of only "attribute-name." A tag with attributes is represented in the format of "<tag-name attribute-name-1=attriute-value attribute-name-2>."

This appendix introduces the common tags required for describing basic HTML, and tags required for calling Web applications. For details of these tags and others, refer to books or Web pages providing an explanation of HTML.

Web Browsers being commonly used are IE (Microsoft's Internet Explorer) which defines its own tags. Therefore, all tags cannot be used for both of these Web Browsers. By the way, even the most common tags may be each interpreted differently by these browsers. In addition, the tag support range or operation depends on the actual version of the Web Browser. For details, refer to related books or Web pages.

The following explains main tags, and attributes available by the tags.

<HTML>~</HTML>

This tag indicates that that the data being sent comprises an HTML document.

<HEAD>~</HEAD>

This tag indicates the header of the HTML document. <TITLE> must be described in the header. <BASE>, <SCRIPT>, <STYLE>, <META>, <LINK>, <OBJECT>, <NEXTID>, and <ISINDEX> can be also described in the header.

<TITLE>~</TITLE>

This tag indicates the title of the HTML document. Generally, the title is not displayed on the page.<BODY>~</BODY>

This tag indicates the body of HTML text. In the body, the text style must be arranged with various tags.

Attribute name	Explanation
BGCOLOR="Color"	Specifies the background color.
BACKGROUND="URL"	Specifies the background image.
TEXT="Color"	Specifies the text color. TEXT indicates the general text, LINK does the link text, VLINK does the cached link text, and ALINK does the link text during mouse clicking.
LINK="Color"	
VLINK="Color"	
ALINK="Color"	

<Hn>˜</Hn>

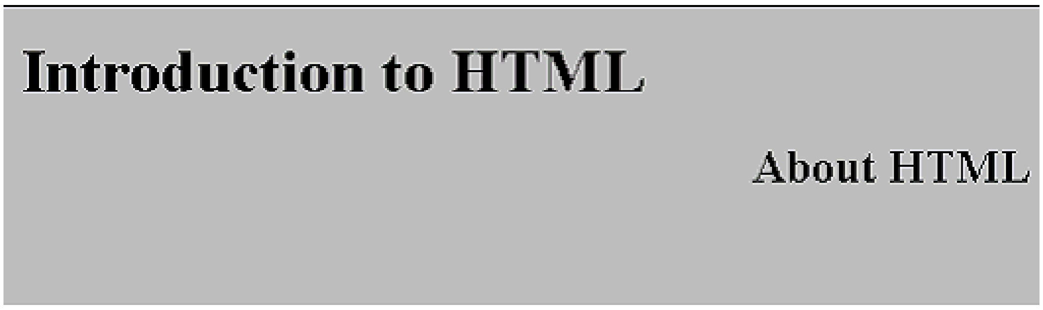
This tag specifies headers such as the title of each chapter. "n" is the header level that can be specified in the range of 1 to 6, with one being the largest and boldest text.

Attribute name	Explanation
ALIGN=Position	Specifies the position where the text is to be displayed. "left," "center," or "right" can be specified as a position.

Example of use

```
<H1> Introduction to HTML </H1>
<H2 ALIGN=right> About HTML </H2>
```

Displayed result



Introduction to HTML

About HTML

<P>˜</P>

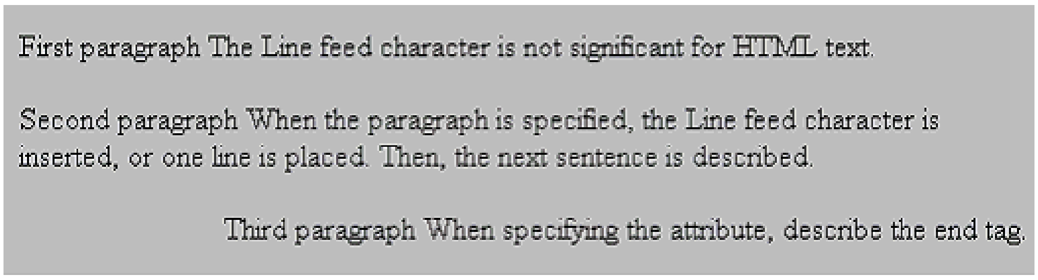
This tag indicates a new paragraph. </P> can be omitted, but is required when specifying any optional attributes.

Attribute name	Explanation
ALIGN=Position	Specifies the displayed position. "left," "center," or "right" can be specified as a position

Example of use

```
<P> First paragraph
The Line feed character is not significant for HTML text.
<P> Second paragraph
When the paragraph is specified, the Line feed character is inserted, or one line is placed.
Then, the next sentence is described.
<P ALIGN=right> Third paragraph
When specifying the attribute, describe the end tag. </P>
```

Displayed result



First paragraph The Line feed character is not significant for HTML text.

Second paragraph When the paragraph is specified, the Line feed character is inserted, or one line is placed. Then, the next sentence is described.

Third paragraph When specifying the attribute, describe the end tag.

<HR>

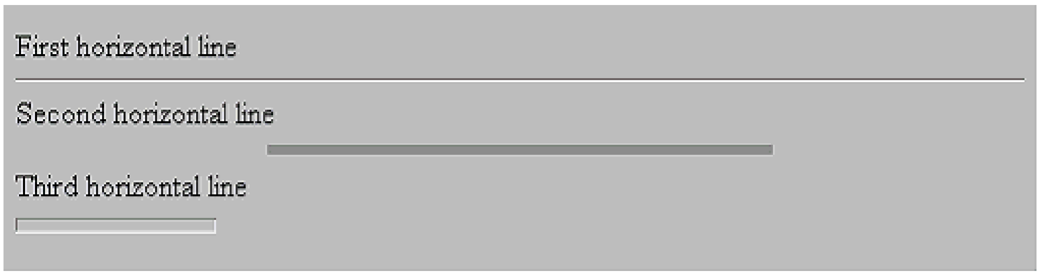
This tag draws a horizontal line.

Attribute name	Explanation
ALIGN=Position	Specifies the displayed position. "left," "center," or "right" can be specified as a position
SIZE=Height	Specifies the horizontal line height in the unit of pixels.
WIDTH=Width	Specifies the horizontal line width in the unit of pixels. For this specification, the rate (%) of the horizontal line width for the Web Browser-displayed width can be used specified.
NOSHADE	Specifies the horizontal line without three-dimensional shading.

Example of use

```
First horizontal line
<HR>
Second horizontal line
<HR SIZE=5 WIDTH=50% NOSHADE>
Third horizontal line
<HR ALIGN=left SIZE=8 WIDTH=20%>
```

Displayed result



<FORM>~</FORM>

This tag indicates an input form. Using the <INPUT> or <SELECT> tag in the form enables the input component to be allocated. When executing the action, the data input to the <INPUT> or <SELECT> tag can be passed, which can be used for executing Web applications.

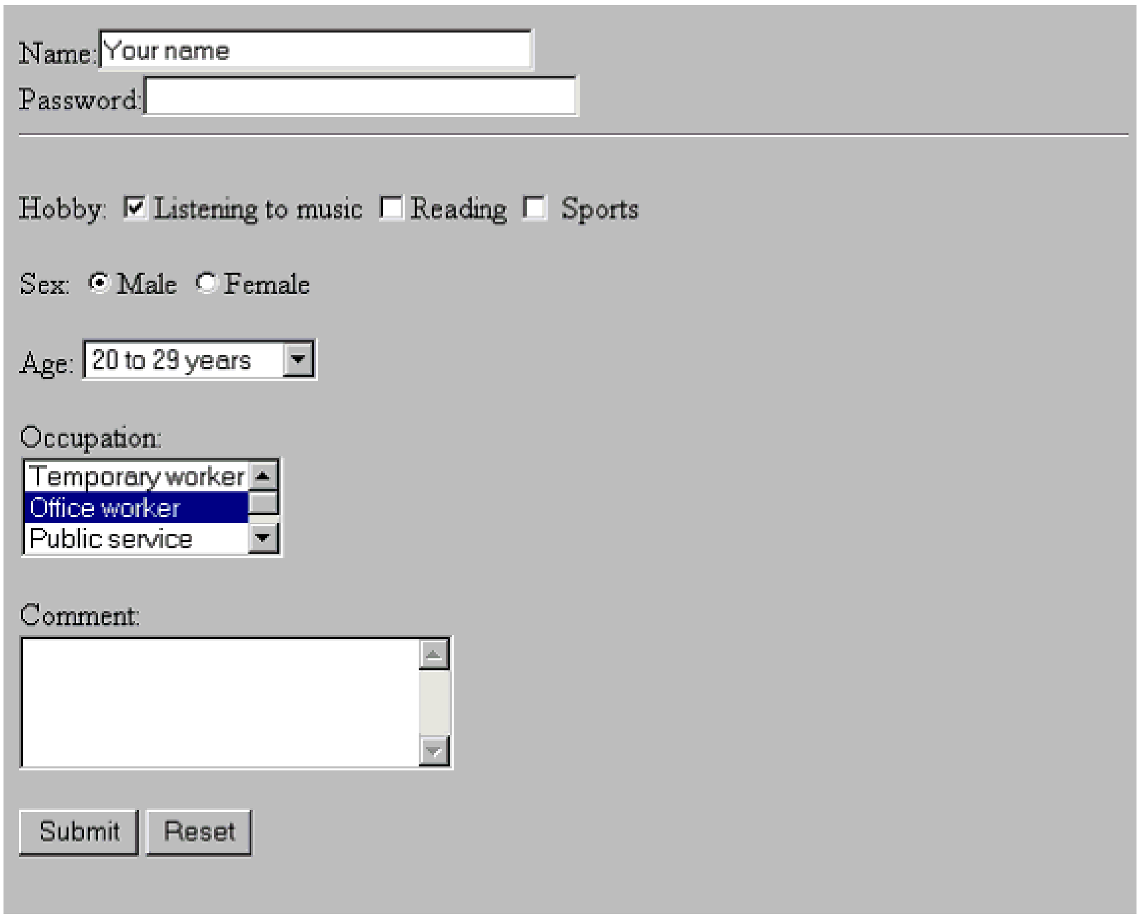
Attribute name	Explanation
ACTION="Action-name"	Specifies the action to be executed when pressing the Submit button. Generally, it specifies a Web application to be executed.
METHOD=Method-name	Specifies the method for passing the data input on the form to the Web application. When the method name is GET, this value is passed to the environment value, or when it is POST, this value is passed to the standard input.
ENCTYPE="Encode type"	Specifies the encode type for data that is input or output to the script. When the file uploading function is used though it is usually an omitted attribute, "Multipart/form-data" is specified as an attribute.
TARGET=Target-name	Specifies the window (frame) where the action execution result is displayed.
NAME=Form-name	Add the name to the form. It is selected and used from the script.
onSubmit="Script"	Specifies the script to be executed when pressing the Submit button.
onReset="Script"	Specifies the script to be executed when pressing the Reset button.

Example of use

```
<FORM METHOD=POST ACTON="sample/action.script">
Name:<INPUT TYPE=TEXT NAME="FULLNAME1" VALUE="Your name" SIZE=30><BR>
```

```
Password:<INPUT TYPE=PASSWORD NAME="PASSWORD1" SIZE=30>

<HR>
<P>
Hobby:
<INPUT TYPE=CHECKBOX NAME="CHECK1" VALUE="Listening to music" CHECKED>Listening to music
<INPUT TYPE=CHECKBOX NAME="CHECK1" VALUE="Reading">Reading
<INPUT TYPE=CHECKBOX NAME="CHECK1" VALUE="Sports"> Sports

<P>
Sex:
<INPUT TYPE=radio NAME="RADIO1" VALUE="Male" CHECKED>Male
<INPUT TYPE=radio NAME="RADIO1" VALUE="Female">Female

<P>
Age:
<SELECT NAME="DRDLIST1">
<OPTION VALUE="10s">10 to 19 years
<OPTION VALUE="20s" SELECTED>20 to 29 years
<OPTION VALUE="30s">30 to 39 years
<OPTION VALUE="40s or later">40 years or later
</SELECT>

<P>
Occupation:<BR>
<SELECT NAME="LIST1" SIZE=3>
<OPTION VALUE="Free">Temporary worker
<OPTION VALUE="Office worker" SELECTED>Office worker
<OPTION VALUE="Public service">Public service
<OPTION VALUE="Self-employed">Self-employed
</SELECT>

<P>
Comment:<BR>
<TEXTAREA NAME="TEXTAREA1" COLS=25 ROWS=4></TEXTAREA>
<P>
<INPUT TYPE=submit VALUE="Submit">
<INPUT TYPE=reset VALUE="Reset">
</FORM>
</BODY>
</HTML>
```

Displayed result



<INPUT>

This tag displays various form components on the input form. The displayed component depends on the TYPE attribute.

Attribute name	Explanation
TYPE=Type-name	The displayed content depends on the type name. TEXT: Text PASSWORD: Password CHECKBOX: Checkbox RADIO: Radio button HIDDEN: Hidden area BUTTON: Button SUBMIT: Submit button RESET: Reset button FILE: File upload
NAME=Name	Specifies the form component name.

<INPUT TYPE-TEXT>

This tag is the form component for inputting the text.

Attribute name	Explanation
SIZE=Width	Specifies the input area width.
MAXLENGTH=Length	Specifies the maximum number of input characters.
onChange=Script	Specifies the script to be called when the input area is changed.
onSelect=Script	Specifies the script to be called when the character string in the input area is selected.
onFocus=Script	Specifies the script to be called when the focus is moved to the input area.
onBlur=Script	Specifies the script to be called when the focus moves out of the input area.

<INPUT TYPE=PASSWORD>

This tag is the form component for inputting the password, which is same as TYPE=TEXT except that the input characters are displayed as asterisks (*).

<INPUT TYPE=CHECKBOX>

This tag is the form component for specifying a checkbox.

Attribute name	Explanation
VALUE="Character-string"	Specifies the value to be sent when this item is checked.
CHECKED	Sets the initial state of this item to the checked state.
onClick=Script	Specifies the script to be called when this item is checked.

<INPUT TYPE=RADIO>

This tag is the form component for a radio button (also called option button in graphical programming).

Attribute name	Explanation
VALUE="Character-string"	Specifies the value to be sent when this item is checked.
CHECKED	Sets the initial state of this item to the checked state.
onClick=Script	Specifies the script to be called when this item is checked.

<INPUT TYPE=HIDDEN>

This tag is for a hidden field and is not displayed on the Web Browser. For example, this field is used for Web applications to inherit the data through Web Browsers.

Attribute name	Explanation
VALUE="Character-string"	Specifies the field value.

<INPUT TYPE=SUBMIT>

This tag executes the action specified by the ACTION attribute in the FORM tag.

Attribute name	Explanation
VALUE="Character-string"	Specifies the character string to be displayed on the button.
onClick=Script	Specifies the script to be called when this item is checked.

<INPUT TYPE=RESET>

This tag resets all input items in FORM to initial values.

Attribute name	Explanation
VALUE="Character-string"	Specifies the character string to be displayed on the button.

Attribute name	Explanation
onClick=Script	Specifies the script to be called when this item is checked.

<INPUT TYPE=BUTTON >

This tag is used for JavaScript, so it is not displayed by Web Browsers that cannot use JavaScript (or JScript).

Attribute name	Explanation
VALUE="Character-string"	Specifies the character string to be displayed on the button.
onClick=Script	Specifies the script to be called when this item is checked.

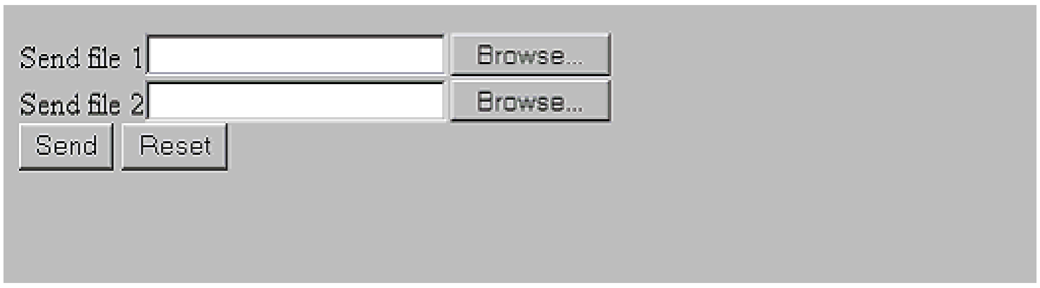
<INPUT TYPE=FILE>

This tag is the form component to specify a file upload from the client side to the Web server side.

Example of use

```
<FORM METHOD=POST ACTON="sample/action.script"
      ENCTYPE="multipart/form-data">
<P>
Send file 1<INPUT TYPE="file" NAME="FILE1"><BR>
Send file 2<INPUT TYPE="file" NAME="FILE2"><BR>
<P>
<INPUT TYPE=submit VALUE="Send">
<INPUT TYPE=reset VALUE="Reset">
</FORM>
```

Displayed result



<TEXTAREA>~</TEXTAREA>

This tag displays the multiple-line-input field on the input form. The text between <TEXTAREA> and </TEXTAREA> is displayed in the field.

Attribute name	Explanation
NAME="Character-string"	Add the name to the field.
ROWS=n	Specifies the number of lines in the text area.
COLS=n	Specifies the number of columns in the text area.
onChange=Script	Specifies the script to be called when the input area is changed.
onSelect=Script	Specifies the script to be called when the character string in the input area is selected.
onFocus=Script	Specifies the script to be called when the focus is moved to the input area.
onBlur=Script	Specifies the script to be called when the focus moves out of the input area.

<SELECT>~</SELECT>

This tag displays the select form component on the input form.

Attribute name	Explanation
NAME="Character-string"	Add the name to the field.
SIZE=n	Specifies the number of displayed lines on the selective field.
MULTIPLE	Can select multiple options.
onChange=Script	Specifies the script to be called when the input area is changed.
onFocus=Script	Specifies the script to be called when the focus is moved to the input area.
onBlur=Script	Specifies the script to be called when the focus moves out of the input area.

<OPTION>~</OPTION>

This tag displays the select item on the SELECT tag. </OPTION> can be omitted.

Attribute name	Explanation
SELECTED	Selects the select state as the initial state.
VALUE="Character-string"	Specifies the value to be sent when this item is checked.

A.5 Notes and Measures Taken regarding Problems with Execution in Web Browsers

For client/server systems, users cannot freely change the control of the application on the client side. For Internet/Intranet applications using Web Browsers, however, the user can easily change the Web Browser's condition, so it may depend on this operation as to whether the consistency of the Web application on the server functions properly.

Generally, problems may be caused by the following operations:

- When the Refresh button of the Web Browser is used.

  This button can re-execute the processing that has already executed, so a conflict may occur in an application, such as during registering or updating.

- When the Submit button (with "SUBMIT" specified in the INPUT tag INPUT attribute) is pressed twice or more times by the user, sending the same transactional request to the Web Server.

  Pressing this button twice or more times causes the same processing to be executed again and again, so a conflict may occur in the application, such as during registering or updating.

- When processing is started in the middle of an application by directly inputting the bookmark or URL.

  An application can be executed without authorization, so a problem regarding security may be caused. In addition, the application is not executed in the correct sequence, so a logic problem may occur.

- When a cached page is referred to.

  Using the cache function of a Web Browser, the requested Web page can be referred to without accessing the server (it is held in cache on the client machine), so the user may refer to secret information contained along with the Web page in the cache, for example: the sales data and personal information.

The following illustrates examples of measures to take against the above noted potential problems, which can be taken by Web Browsers or Web applications on the server.

User's operation	Examples of measures	
	Web Browsers	Web applications
Use the Refresh button of the Web Browser.	Using JavaScript, open the window without the Return button.	Keep track of the application execution sequence, and confirm that the application is correctly executed.
Press the Submit button twice or more times.	Using JavaScript, determine whether the application is half-processed. If the application is half-processed, disable the Submit button.	Design Web applications not to cause a problem even when the same processing is executed twice or more times.
Start the processing in the middle of a business by directly inputting the bookmark or URL.	Using JavaScript, open the window without a bookmark.	Keep track of the application execution sequence, and confirm that the application is correctly executed.
Refer to the cached page.	Invalidate the cache of the Web Browser.	Keep track of the application execution sequence, and confirm that the application is correctly executed.

Generally, the measures to take depend on the content of an application. For example, the cache may not need to be invalidated for applications with low security requirements. If another operational problem is caused by the content of the application, take the measures against the problem from its own aspect.

Appendix B Use of Unicode

All COBOL Web subroutines in NetCOBOL support Unicode. Web applications that operate using Unicode improve the expressional performance as the number of characters increases. However, these Unicode Web applications differ from conventional Web applications that use native code. Note the following when creating and executing Web applications that support Unicode.

The following table indicates the differences to consider when creating and executing Web applications using native code and Web applications using Unicode.

No.	Item	Native code	Unicode
1	WWW browsers	WWW browser that displays native code. Any of the latest WWW browsers can be used	WWW browser that supports UTF-8 or UCS-2 and transmits Web parameters on UTF-8.
2	Input code system from WWW browser	Native code	UTF-8 (*1)
3	Output code system to WWW browser	Native code	UTF-8 or UCS-2
4	Compile options	Not required	Required
5	Unicode supported subroutines	Partial limitations	Partial limitations
6	HTML document code system for Web pages used for invoking applications	Native code	UTF-8, or UCS-2
7	HTML document code system for Web pages used for processing output result (prototype file)	Native code	UTF-8, UCS-2, or native code
8	Log file code system	Native code	UCS-2 or native code
9	Coexistence of NetCOBOL applications with different code systems	Basically, code systems cannot be mixed. However, it is possible under certain conditions depending on the subroutines and WWW servers.	

*1 : Any input code system can be used in a COBOL application because the NetCOBOL Web subroutines execute conversion processing according to the COBOL application code system. Each case is described in the following topics.

B.1 WWW Browsers

Native code Web applications can be used with any WWW browser. However, Unicode Web applications can only be used with a WWW browser that supports Unicode. Note that old WWW browsers in particular do not typically support Unicode. Also, the latest WWW browsers support Unicode at various levels. When Unicode is used in a NetCOBOL Web subroutine, the WWW browser must be able to do the following:

- Display the UTF-8 or UCS-2 HTML document correctly, and
- Transmit the Web parameter that is transmitted from a WWW browser using UTF-8.

B.2 Input Code System From WWW Browser

Native code Web applications automatically identify whether the input code system from the WWW browser is native code. Therefore, the HTML document code system used for Web pages for invoking an application could be native code. However, a Unicode Web application could only identify UTF-8. This is because Unicode Web applications cannot identify UTF-8 and other code systems automatically and because in general WWW browsers transmit Web parameter code systems as UTF-8 when Unicode HTML documents are used. Therefore, a Web page for invoking an application that activates a Unicode Web application must be in Unicode.

B.3 Code System Output to WWW Browser

In a conventional native code Web application, the code system output to the WWW browser is native code. However, in a Unicode Web application, the code system can be UTF-8 or UCS-2. UTF-8 is selected to decrease the data amount when many ASCII characters are used throughout the system, and UCS-2 is selected to decrease the data amount when many national characters are used. The code system output to the WWW browser is selected in the following environment variable information:

@CBR_WEB_OUT_CODE

Specifies the code system output to the WWW browser.

Value	Meaning
<u>UTF8</u>	Specifies UTF-8 as the code system output to the WWW browser.
UCS2	Specifies UCS-2 little-endian as the code system output to the WWW browser.

Note

- The output code system of a native code Web application is fixed to native code. Therefore, no value can be specified in this environment variable.
- When this environment variable information is set by the Web application itself, the results are undefined.

B.4 Compile Options

A native code Web application does not require any special compile options. However, a Unicode Web application requires RCS (UCS2) to be specified as a compile option.

B.5 Unicode Supported Subroutines

The Web subroutines listed in the table below cannot be used by Web applications with Unicode. The alternative subroutines listed must be used with Unicode.

Subroutine name	Alternative subroutine
COBW3_GET_VALUE	COBW3_GET_VALUE_XX COBW3_GET_VALUE_NX COBW3_GET_VALUE_XN COBW3_GET_VALUE_NN
COBW3_CHECK_VALUE	COBW3_CHECK_VALUE_X COBW3_CHECK_VALUE_N
COBW3_SET_CNV	COBW3_SET_CNV_XX COBW3_SET_CNV_NX COBW3_SET_CNV_XN COBW3_SET_CNV_NN
COBW3_DEL_CNV	COBW3_DEL_CNV_X COBW3_DEL_CNV_N
COBW3_SET_REPEAT	COBW3_SET_REPEAT_XX COBW3_SET_REPEAT_NX COBW3_SET_REPEAT_XN

Subroutine name	Alternative subroutine
	COBW3_SET_REPEAT_NN
COBW3_DEL_REPEAT	COBW3_DEL_REPEAT_X COBW3_DEL_REPEAT_N
COBW3_SET_COOKIE	COBW3_SET_COOKIE_XX COBW3_SET_COOKIE_NX COBW3_SET_COOKIE_XN COBW3_SET_COOKIE_NN
COBW3_DEL_COOKIE	COBW3_DEL_COOKIE_X COBW3_DEL_COOKIE_N
COBW3_GET_COOKIE	COBW3_GET_COOKIE_XX COBW3_GET_COOKIE_NX COBW3_GET_COOKIE_XN COBW3_GET_COOKIE_NN

Table B.1 The conventional subroutines

Subroutine name	Alternative subroutine
COBW3_NAME	COBW3_GET_VALUE_XX COBW3_GET_VALUE_NX COBW3_GET_VALUE_XN COBW3_GET_VALUE_NN
COBW3_VALUE	COBW3_CHECK_VALUE_X COBW3_CHECK_VALUE_N
COBW3_CNV_SET	COBW3_SET_CNV_XX COBW3_SET_CNV_NX COBW3_SET_CNV_XN COBW3_SET_CNV_NN
COBW3_CNV_DEL	COBW3_DEL_CNV_X COBW3_DEL_CNV_N
COBW3_CNV_INIT	COBW3_INIT_CNV

Refer to the associated Subroutines User's Guide for the for details.

B.6 HTML Document Code System for Web Pages Used for Invoking Applications

In a native code Web application, the document code systems used for a Web page that invokes an application are native code. However, in a Unicode Web application, the document code systems used for a Web page that invokes an application must be UTF8 or UCS-2 (depending on the type of WWW browser).

B.7 HTML Document Code System for Web Pages Used for Processing Result Output

In a native code Web application, the only document code system that can be used for a Web page that processes output results is native code. However, the following HTML document code systems can be used in a Unicode Web application:

- Native code
- UTF-8
- UCS-2

It is recommended that the code systems for HTML documents are the same as the output code systems for the WWW browser. This will improve the execution performance.

B.8 Code System for Log File

The code system for the log file that is output by the Web subroutines is determined as described below:

When creating a new log file:

The first COBOL application code system referenced in the execution environment is used.

When adding log information to an existing log file:

The existing log file code system is used.

B.9 NetCOBOL Applications with Combined Code Systems

NetCOBOL applications with different code systems cannot be used in the same process. Therefore, it is not possible to use multiple Web application code systems in IIS and NES. However, in IIS, multiple code systems can be used within the same WWW server when separate processes are used. In a CGI application, multiple code systems can also be used within the same server, because each process is operated independently. Refer to the *NetCOBOL User's Guide* for details of combined code systems.