

VisualAge Pacbase



# Dialog Web Revamping Operator's Guide

*Version 3.5*



**Note**

Before using this document, read the general information under "Notices"

According to your license agreement, you may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:

<http://www.ibm.com/software/awdtools/vapacbase/productinfo.htm>

Consult the Catalog section in the Documentation home page to make sure you have the most recent edition of this document.

2nd Edition (April 2005)

This edition applies to the following licensed program:  
VisualAge Pacbase Version 3.5

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at:

<http://www.ibm.com/software/awdtools/vapacbase/productinfo.htm>

or to the following postal address:

IBM Paris Laboratory  
Support VisualAge Pacbase  
1 place J.B. Clément  
93881 Noisy-le-Grand Cedex FRANCE

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1983, 2003. All rights reserved.

Note to U.S. Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Contents

## Foreword .....5

## Chapter 1: General Presentation .....7

Dialog Web Revamping.....7

How to Generate a J2EE Web Application from a TUI Application.....7

## Chapter 2: Running an Application via a Web Browser .....9

General Principle.....9

Middleware.....10

## Chapter 3: Extracting and Transferring Files.....11

Extracting Screens from the Repository .....11

Transferring the Extraction File to a Workstation .....11

## Chapter 4: Launching the generator .....13

Standalone version.....13

Plug-in version on WSAD .....13

## Chapter 5: Revamping an Application .....14

Graphic Interface of the Generator .....14

    Required Parameters.....14

    Generation Options .....16

        Communication .....17

        Presentation Parameters .....19

        Various Parameters .....21

        Keys and Action Command.....22

        Icons.....23

Command Lines Interface .....24

    Starting the Generator.....24

    Modifying Generation Parameters .....24

Generation Output.....27

## Chapter 6: Presentation of Generated Pages .....31

First Page.....31

Structure of the HTML Page .....32

    Screen Description .....33

        Head.....33

        Body .....34

Structure of JSP Pages .....34

Structure of Help Pages .....35

    Contextual Help .....36

    General Help (summary).....36

    The Navigation Bar.....36

## Chapter 7: Customizing the Generated Templates .....37

Styles .....37

Automatic Modification of HTML Pages .....38

JavaScript Functions.....39

    Presentation of Javascript Functions .....39

        Functions used at General Level.....39

        Functions used at Dialog Level.....41

        Functions used at Screen Level .....41

    Using JavaScript Functions as Entry Points .....42

Templates.....43

    Presentation of Templates.....43

        First Page .....43

        Common Page.....43

    Screen Page.....44

        Help Page .....44

    Template Description .....44

    Modifying the Templates.....46

## Chapter 8: Appendix..... 47

Configuration of a Servlet Container .....47

    Initialization Parameters of the PacWebServlet Servlet49

Migration of Applications (Versions 2.5, 3.0) towards J2EE

Mode (Version 3.5) .....50

    Calling the Servlet .....50

    Managing the Cookies for the User/Password .....51

    Adding the New Files.....51

    Adding the RootContext in Pages .....51

    VAP Middleware Configuration File.....51

Example of the Use of Conversation Proxies .....52

Use with 'CICS Transaction Gateway' .....53

## NOTICES

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Intellectual Property and Licensing  
International Business Machines Corporation  
North Castle Drive, Armonk, New-York 10504-1785  
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of information which has been exchanged, should contact:

IBM Paris Laboratory  
Département SMC  
1 place J.B. Clément  
93881 Noisy-le-Grand Cedex  
FRANCE

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may change this publication, the product described herein, or both.

## TRADEMARKS

IBM is a trademark of International Business Machines Corporation, Inc. AIX, AS/400, CICS, CICS/MVS, CICS/VSE, COBOL/2, DB2, IMS, MQSeries, OS/2, PACBASE, RACF, RS/6000, SQL/DS, TeamConnection, and VisualAge are trademarks of International Business Machines Corporation, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

All other company, product, and service names may be trademarks of their respective owners.

# Foreword

## Typographical conventions in use

The **courier New** font is used for any character string to be entered, displayed or corresponding to generated code, as well as for the name of files, directories, windows, menu or components.

*Italics* is used for titles of publications and chapters in cross-references.

The following icons are used:



note, remark, important point,



cross-reference to another location in the documentation or to another documentation,



Important, caution.

## Terminological conventions in use

The terms **user** and **developer** are used in the *Operator's Guide* with the following meanings:

- the **user** is the person who uses the final Web application. He/she works on a personal computer on which a browser is installed.
- the **developer** is the person in charge of installing Dialog Web Revamping, generating the revamping parameter files, and installing the final application on the users' computer.

The term **screens** is used for screens in an application developed with the On-line System Development generator.

The term **page** is used for HTML or JSP pages generated with Dialog Web Revamping.

The term **screen page** is used for a page describing a screen.



## Chapter 1: General Presentation

### Dialog Web Revamping

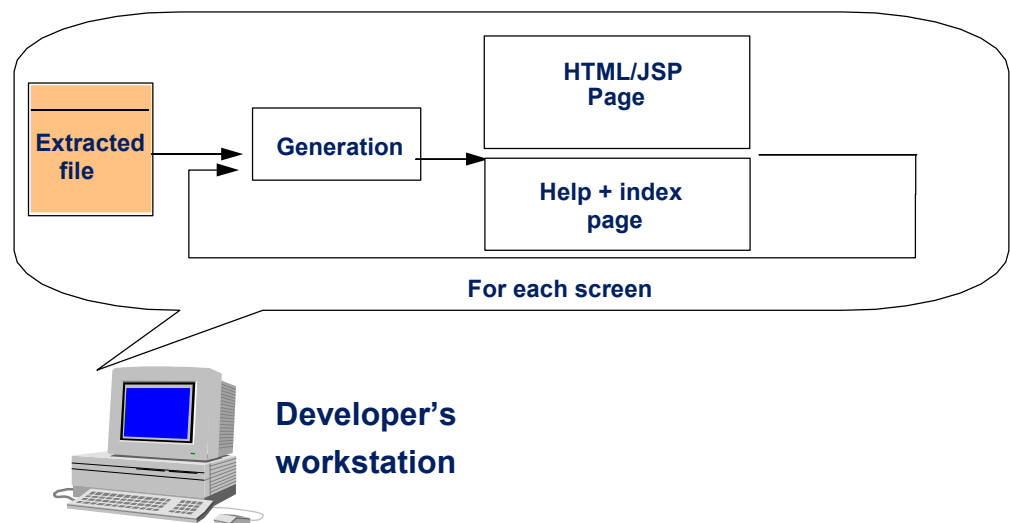
Dialog Web Revamping is designed to automatically generate J2EE Web applications from your existing VisualAge Pacbase applications at a low cost. It offers both an opening to Internet/Intranet and a greater use comfort (mouse-support, drop-down lists and interactive help).

With its numerous generation options, screens generated as HTML or JSP pages can be entirely customized both at the user interface level (addition of push buttons, etc.) and at the processing level (addition of field controls, etc.) by using Java language (Javascript, Applets, etc.) or by using any HTML editor.

### How to Generate a J2EE Web Application from a TUI Application

The main steps consist in:

- extracting, from the VisualAge Pacbase Repository, the files containing the description of your screens and transferring them onto the developer's workstation,
- setting communication parameters,
- defining options for the revamping of the On-Line Systems Development application and generate HTML templates or JSP, index and help pages.







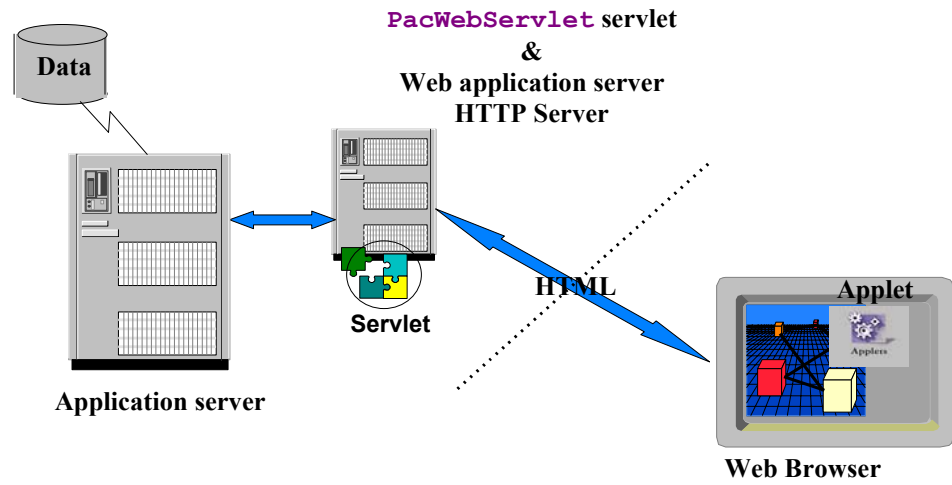
## Chapter 2: Running an Application via a Web Browser

### General Principle

When it is used via a Web browser, the application has the following operating mode:

The user requests an access to the application by selecting a link in an HTML first page. This link activates the **PacWebServlet** servlet which:

- sets up a context for the management of the user's conversation and initializes a session if it is the first request, or retrieves an existing context if it is not the first request.
- sends the request to the transactional application through the communication layers and transactional monitor.
- The application sends a logical message which corresponds to the requested screen. The message includes general information related to the conversation context or to the screen (conversation identifier, session number, screen code, library used, database code...). The other part of the logical message includes all the information you find on the screen (labels, variable fields, ...) and their codes (location in the screen, line, column, length, nature, intensity, color, Data Element code, ...).
- The logical message is then decoded by the 'proxy conversation' layer.
- For HTML, the servlet will then merge the data with the generated HTML templates.
- For JSP, the servlet transmits to the generated JSP by sending it the data.
- If a page has not been generated, the servlet will generate this page 'on the fly' in the HTML format.
- The HTML page which results from this merging is sent to the user via the HTTP server.



## Middleware

The servlet and the mainframe communicate via the VisualAge Pacbase middleware. The Middleware type and parameters must be chosen upon generation.

For more information on the use of VA Pac middleware, read the *Middleware User's Guide*.

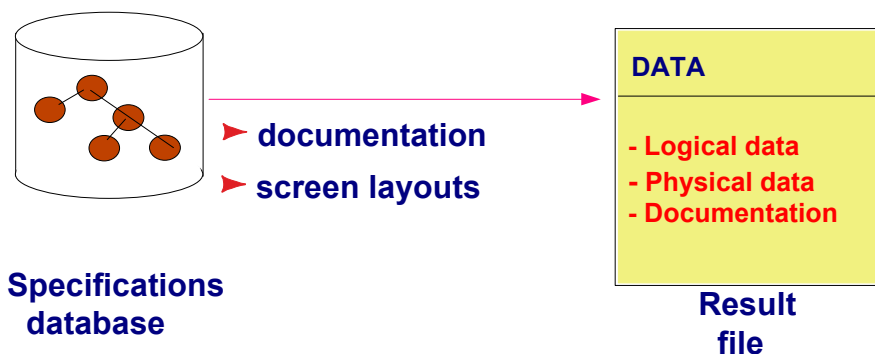
## Chapter 3: Extracting and Transferring Files

VA Pac Repository generator generates TUI application screens and creates files which contain the screen descriptions. These files must be extracted, they are then used to create HTML templates or JSP, which display the contents of these screens in a Web navigator.

### Extracting Screens from the Repository

You must first extract, from the VA Pac Repository, the data required for local parameterizing :

- From the Dialog: application title, options, documentation
- From the Screen: definition, description, documentation
- From the Data Elements: name, code, length, authorized values, documentation
- Dialog help



The file which contains the descriptions of the screens to be revamped is output by the VA Pac **GPRT** (**GEO** option **C4**) procedure. The procedure output is a user file: **PAC7GT** (or **GT** for GCOS8 system) whose records are each 180-characters long (maximum).



For more information on the **GPRT** procedure, see the *Character-Mode User's Interface Guide*.

### Transferring the Extraction File to a Workstation

Downloading the **PAC7GT** or **GT** file onto a workstation is performed by the developer. The transfer utility must be parameterized so as to keep all special characters. The local resulting file of the transfer is what we will name here **FILENAME.EXT**.



The modification of the On-Line Systems Development programs for the implementation of Dialog Web Revamping is documented in the *On-Line Systems Development Reference Manual*.

## Chapter 4: Launching the generator

There are two versions of Dialog Web Revamping Generator, the standalone version and the plug-in version for WebSphere Studio (WSAD).

### Standalone version

To launch the Generator, execute the `vapWebGen.bat` file, either directly from the Files Manager, or via a shortcut on your desktop.

### Plug-in version on WSAD

To launch the Generator from WSAD, follow these steps:

- Open WebSphere Studio,
- Select a Web perspective,
- Click in the Toolbar on 'Files' → "New" → 'Other' to create a Web project, or right-click on an existing Web project, and select 'New' → 'Other',
- Select 'VisualAge Pacbase eBusiness' → 'Dialog Web Revamping Generator' and click 'Next'. This action displays the Generator's first window.

## Chapter 5: Revamping an Application

This step consists in defining the options and parameters which will be used to revamp your VisualAge Pacbase application. Then the generator automatically transforms the application screens into HTML templates or JSP and the selected parameters are saved in a file. You can choose the revamping parameters and launch the generator from a graphic interface or a textual interface.



You are presented, just below, the windows of a generator from a standalone version. The fields displayed in the windows of the plug-in version of the generator are sometimes presented differently and their names can be different but their functions are the same.

### Graphic Interface of the Generator

The parameter files which enable the application revamping are generated by the generator, which handles data extracted from the database. This revamping step is automatic and only requests a very small participation from the developer.

To start the generator, you must execute `vapWebGen.exe`, either directly from the file manager, or via a shortcut.

### Required Parameters

The screenshot shows a dialog box titled "Dialog Web Revamping Generator". The main area is titled "Required parameters". It contains a file list area with buttons "Add File(s)", "Deselect File(s)", and "Deselect All". Below this are input fields for "Generation directory", "Application Name", "Short Application Name" (containing "APPN"), "HTML" (a dropdown menu), and "Context Path" (containing "/pacweb"). There are also checkboxes for "Generate only .INI file" and "Enabled security", both of which are checked. At the bottom, there are buttons for "<<Back", "Next>>", "Finish", and "Cancel". A status bar at the bottom left says "Generation Directory required".

- **Extraction files** panel: : Select the extraction file(s) from which you will generate your HTML templates or your JSP.
  - With the **Add File(s)** button, you can add one or more files in the list.
  - With the **Deselect File(s)** button, you can deselect one or more files.
  - With the **Deselect all** button, you can deselect all the files.
- **Generation directory** : Enter the directory in which the file structure tree will be generated. For example, if the generation directory is **c:\pacweb2** and the application short name is **WE**, you will have:
  - **c:\pacweb2\WE\_app**: directory which contains the HTML templates of the application.
  - **c:\pacweb2\WE\_help**: directory which contains the HTML pages of the application help.
  - **c:\pacweb2\icon**: directory which contains the icons displayed in the HTML templates.
  - **c:\pacweb2\WE\js** : directory which contains JavaScripts function files.

The **browse** button enables you to select the generation directory.

- **Application Name**: Enter the name which will be used for the:
  - **<Application name>.ini** file in which the generation parameters will be saved (e.g. the Middleware parameters used for the application).
  - **<Application name>.war** file which corresponds to the web application in a J2EE Web Archive format which makes it possible to deploy a J2EE application.
- **Short Application Name**: character string which differentiates the generated applications. This character string is used to create the various directories of each application.
- **HTML/JSP**: Choose the type of the generated pages:
  - **HTML**: in this case, the code which merges the data is present in the servlet.
  - **JSP**: in this case, the code which merges the data is present in the JSP page in the form of a scriptlet.
- **Context Path**: Enter here the root of the application deployed on a web server (useful for HTML generation only).
- **Generate only .INI file**: If you check this box, only the file which contains the parameters is generated.
- **Enabled security**: check this box to apply access security to the Web application generated by Dialogue Web Revamping generator.

In WebSphere Studio, an interface allows you to define user groups and to manage profiles.

- ☞ To implement access security from WebSphere Studio:
  - ❑ Import in WebSphere Studio, the Web file generated by Dialogue Web Revamping generator, **<NomApplication>.war**,
  - ❑ Create a server and a server configuration for your Application,
  - ❑ create one or more security roles and security constraints, and indicate authorizations for your security role(s) To do so, complete the Web Application Deployment Descriptor file (application.xml) and the EAR Deployment Descriptor file.
  
- ☞ For more details, refer to IBM Internet documentation, especially the Redbook on security '*IBM WebSphere Security, WebSphere Handbook Series*' and WebSphere Studio's on-line help.

## Generation Options

These options enable you to customize your pages. You can select:

- The communication type (via a middleware, a gateway or a CICS transaction), as well as the parameters specific to each communication type.
- The Presentation parameters (error messages, background color, fields...).
- Various parameters (suffix of the files which contain the HTML pages, name of the first HTML page, application title, local controls, generation of comments).
- The names of the icons in the menu bars.
- The labels associated with the keys and action commands.

All these generation options are saved in a file named **<application\_name>.ini**. They can be used again for a re-generation. The data required for the operation of the **PacWebServlet** sevlet is generated in the **web.xml** file.



## Communication

Dialog Web Revamping Generator

Communication

MIDDLEWARE

Adapter  GATEWAY

CICS TRANSACTION

Location File  Browse

Location  Load

Code page file  Browse

Location file required <<Back Next>> Finish Cancel

Dialog Web Revamping Generator

Communication

MIDDLEWARE

Adapter  GATEWAY

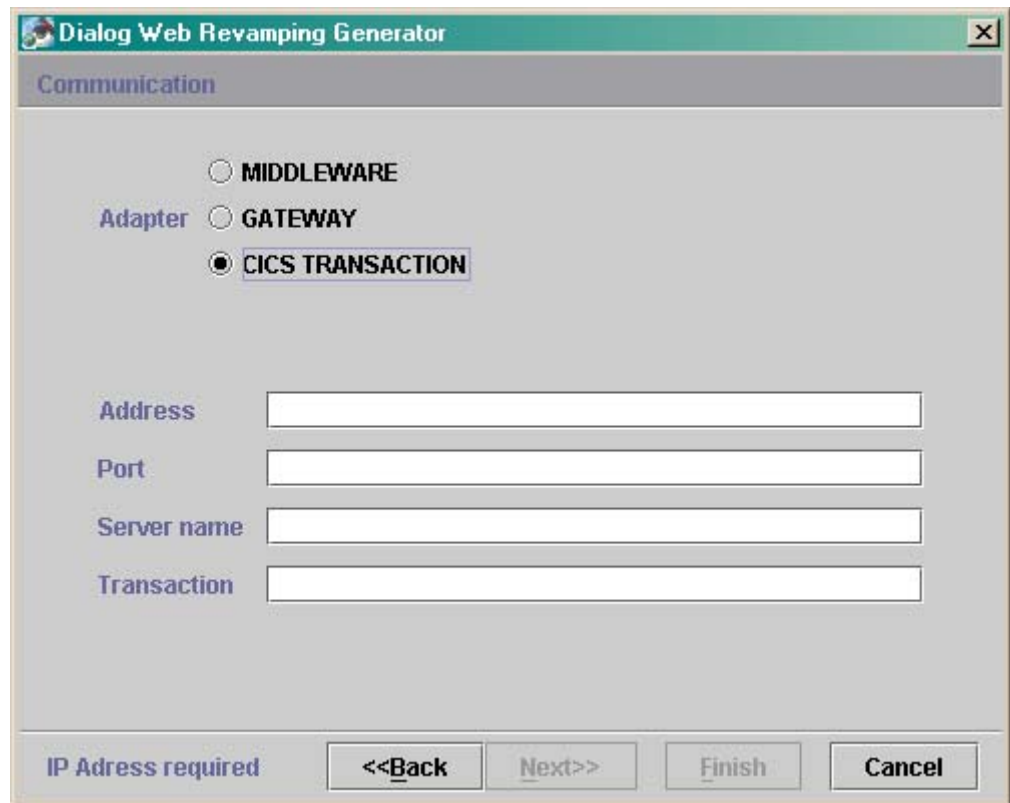
CICS TRANSACTION

Address

Port

Location  Load

<<Back Next>> Finish Cancel



- **Adapter:** Select here the adapter used to communicate with the host application:
  - **Middleware:** direct middleware (in this case, the dlls of the middleware must be accessible via the **PATH** environment variable)
  - **Gateway:** use of a gateway
    - ☞ Refer also to the *eBusiness Applications – Proxy Programming Interface Guide*, chapter *Attributes*, subchapter *Management of Communications*, section *Selection of the Communication Adapter*.
  - **CICS Transaction :**
    - ☞ See also the *Appendix*, subchapter *Use with 'CICS Transaction Gateway'*.

If you select **Middleware**

- **Location file:** the file which contains the locations.
- **Location:** the location in use.
- **Code page file:** the transcoding file used by the middleware.

If you select **Gateway**

- **Address**: the TCPIP address of the machine on which the gateway runs.
- **Port**: the port number on which the gateway is listening.
- **Location**: the location in use.

If you select **CICS Transaction Gateway** :

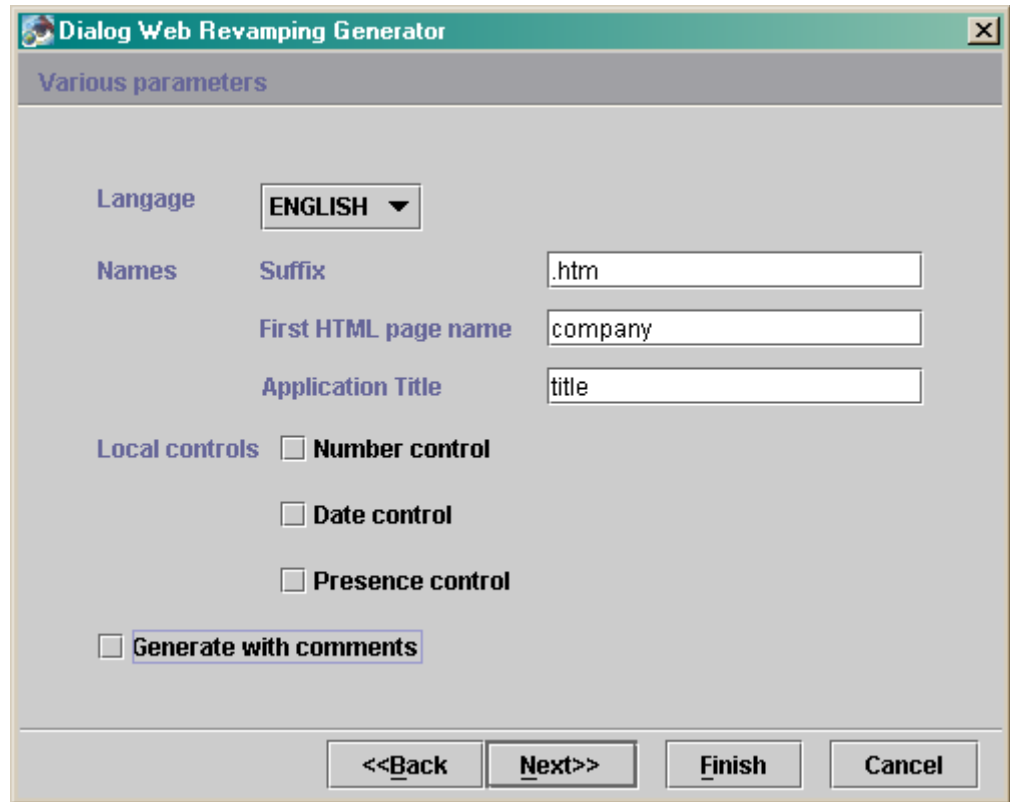
- **Address**: the TCPIP address of the machine on which the gateway (CTG) runs.
- **Port**: the port number on which the gateway (CTG) is listening.
- **Server name**: the server name for CTG.
- **Transaction**: the transaction name.

### Presentation Parameters

- **Error message in**: with this option, you can parameterize the display of errors.
  - **Javascript window**: the error message is displayed in a window, directly in the browser.

- **Field:** the message is displayed at the bottom of the page as indicated in the layout.
- **Both:** the message is displayed in a window and at the bottom of the page simultaneously.
- **Background:** select the background color of the page.
  - **Is a color:** select a color in the drop-down list.
  - **Is a file:** select a bitmap in the: `<ShortApplicationName>\icon` directory (`<ShortApplicationName>` is the name entered in the **Short Application Name** field of the **Required parameters**).
- **Fields:** variable part of the screen
  - **Use DIALOG settings:** use of the display as it is described in VisualAge Pacbase.
  - **Use Style sheet:** use of the `.libvar` style which is described in the `style.css` file. This file is located in the `<ShortApplicationName>\js` directory.
- **Labels:** fixed part of the screen.
  - **Use DIALOG settings:** use of the display as it is described in VisualAge Pacbase.
  - **Use Style sheet:** use of the `.libfix` style which is described in the `style.css` file. This file is located in the `<ShortApplicationName>\js` directory.
- **Labels for list and Radio button:**
  - **Short:** use the short label as the value displayed in a drop-down list or a radio-button.
  - **Long:** use the long label as the value displayed in a drop-down list or a radio-button.
- **Radio button to list limit:** this value is the bound from which a selection list is displayed in a drop-down list rather than in radio-buttons.
- **Other screen links:** enables you to display the links between the application screens (drop-down list, button or edit box).

## Various Parameters



The screenshot shows a dialog box titled "Dialog Web Revamping Generator" with a sub-header "Various parameters". The dialog contains the following fields and options:

- Language:** A dropdown menu set to "ENGLISH".
- Names:**
  - Suffix:** A text box containing ".htm".
  - First HTML page name:** A text box containing "company".
  - Application Title:** A text box containing "title".
- Local controls:** Three unchecked checkboxes labeled "Number control", "Date control", and "Presence control".
- Generate with comments:** An unchecked checkbox.

At the bottom of the dialog are four buttons: "<<Back", "Next>>", "Finish", and "Cancel".

- **Language:** this option enables you to generate the first page and the standard messages of local controls in French or in English. The first HTML page contains a title and the buttons which start the generated applications.
- **Names:**
  - **Suffix:** suffix used in HTML file names (.htm ou .html)
  - **First HTML page name:** name of the first HTML page
  - **Application title:** title which will be displayed in the start button of the application, in the first page.
- **Local controls:**
  - **Number control.**
  - **Presence control.**
  - **Date control.**

These functions are described in the `public.js` file located in the `js` directory.

- **Generate with comments:** adds comments in the generated pages to enable a postprocessor to modify these pages.

## Keys and Action Command

The screenshot shows a dialog box titled "Dialog Web Revamping Generator" with a sub-header "Keys and Action Command". It contains two main sections: "Edit Commands" and "Keys".

**Edit Commands:** This section has four rows, each with a label and a text input field:

- Create text:** Input field contains "Create".
- Modify text:** Input field contains "Modify".
- Cancel text:** Input field contains "Cancel".
- Create or Modify text:** Input field contains "Create or Modify".

**Keys:** This section has several options:

- Standard help PFKEY:** A checkbox that is currently unchecked.
- location of PFKeys:** A dropdown menu set to "TOP".
- PFKeys:** A dropdown menu set to "PFKEY1" and an adjacent text input field containing "PF1".
- Other Keys:** A dropdown menu set to "ENTER" and an adjacent empty text input field.

At the bottom of the dialog are four buttons: "<<Back", "Next>>", "Finish", and "Cancel".

- **Edit Commands:** here you indicate the labels which will represent the action codes (Create, Modify, Cancel, Create or Modify) in selection lists.
- **Standard help PFKEY:** when this box is checked, the function keys used to activate the standard help and the standard contextual help, are mapped as buttons.  
Standard help is directly generated by the application server unlike the HTML pages help which is generated from the extracted file.
- **Location of PFKEYs:** enter here the location of function keys in the screen: on the top or bottom of the screen or in the footer.
- **PFKEYs:** you may enter labels which will be displayed in the buttons associated with function keys on the top or bottom of the screen. You may insert up to 12 keys.
- **Other keys:**
  - **Enter:** Label of the button associated with the Enter key.
  - **Clear:** Label of the button associated with the Clear key.
  - **Pg Up:** Label of the button associated with the Page Up key.
  - **Pg dn:** Label of the button associated with the Page down key.

## Icons



The screenshot shows a dialog box titled "Dialog Web Revamping Generator" with a sub-header "Icones". It contains several input fields for defining icons:

- Help**: Icon name:  associated text:
- Exit**: Icon name:  associated text:
- index**: Icon name:  associated text:
- Backward**: Icon name:  associated text:
- Forward**: Icon name:  associated text:
- Icon next to the title**: Icon name:


At the bottom, there are four buttons: "<<Back", "Next>>", "Finish", and "Cancel".

With these options:

- you name the files which contain the icons of the standard buttons. You can modify all displayed filenames.
- and you can enter the text of the default label associated with the icon in the **Associated text** field.

These icons are contained in the **icon** directory of the application.

- **Help**: enables the user to view help pages, the help on help and the help index.

 For more details, see Chapter *Presentation of Generated Pages*, Subchapter *Structure of Help Pages*.

- **Exit**: enables the user to quit the application or help pages.
- **Index**: enables the user to display the index again.
- **Backward**: enables the user to navigate in the help pages he/she has viewed already. By clicking on this button, he/she will view the previous page.
- **Forward**: enables the user to navigate in the help pages he/she has viewed already. By clicking on this button, he/she will view the next page.
- **Icon next to the title**: name of the file which contains the icon to be inserted at the top of each page.

## Command Lines Interface

### Starting the Generator

To start the generator via a command line, you must enter the following command:

```
VapWebGenBatch.bat <generation_dir> <parameters_file>  
<Geofile>
```

**generation\_directory** refers to the directory where all the HTML pages are generated.

**Parameters\_file** refers to the file which contains the generation parameters. The default file used is **PacWebgen.ini**.

**Geofile** refers to the file resulting from the extraction of the VisualAge Pacbase files which contain the screen descriptions (**GEO C4**).

### Modifying Generation Parameters

The **Pacwebgen.ini** file is used by default to generate an HTML application. You must fill in some of the fields in the file before starting the generation.

The generation parameters are:




Default value	Other possible value	Description
[Controls]		
Number control= <b>NO</b>	<b>YES</b>	Number control
Date control= <b>NO</b>	<b>YES</b>	Date control
Presence control= <b>NO</b>	<b>YES</b>	Presence control
[Format]		
Label format= <b>STYLE</b>	<b>DIALOGUE</b>	<b>DIALOGUE</b> : use of display as it is described in VisualAge Pacbase <b>STYLE</b> : use of a style sheet
Field format= <b>DIALOGUE</b>	<b>STYLE</b>	<b>DIALOGUE</b> : Use of the display format as it is described in VisualAge Pacbase <b>STYLE</b> : use of a style sheet
Radio to list= <b>2</b>		This value is the bound from which a selection list is displayed in a drop-down list rather than in radio-buttons
Comment= <b>NOCOMMENT</b>	<b>COMMENT</b>	Generation with or without comments
SLabel= <b>short</b>	<b>long</b>	Type of label displayed in a drop-down list or a radio button.
[application]		
First page name= <b>company</b>		First page name
Suffix= <b>.htm</b>	<b>.html</b>	Suffix used for HTML pages
Language= <b>ENGLISH</b>	<b>FRENCH</b>	Generation language
ContextPath= <b>/pacweb</b>		Root of the web application for an HTML generation
Create_War_File= <b>YES</b>	<b>NO</b>	Enables you to inhibit the creation of the <b>.war</b> file
GenerationType= <b>HTML</b>	<b>JSP</b>	Generation type
Short application title= <b>APPN</b>		Character string used to create the various directories of each application.
Application title= <b>title</b>		Title displayed in the start button in the first page
Instal_Dir=		Name of the directory which contains the generated templates
Error message= <b>BOTH</b>	<b>WINDOW</b> or <b>FIELD</b>	Enables you to specify whether the error message will be displayed in a window, at the bottom of the page, or both simultaneously.
ShortInstal_Dir		Name of the generation directory

Default value	Other possible value	Description
[Controls]		
[middleware]		
EPIConversationFactory.transactionId		Transaction in CTG-mode
EPIConversationFactory.geoFile		Names of the extraction files
EPIConversationFactory.address		Address of the machine used by the CTG gateway
EPIConversationFactory.server		Name of the server for CTG
EPIConversationFactory.port		Port number for the gateway
ConversationFactory	<code>com.ibm.vap.webconnection.ctg.EPIConversationFactory   com.ibm.vap.webconnection.pwcm.PacWebConversationFactory  </code>	
PacWebConversationFactory.serverAdapter.codePageFile		Transcoding file
PacWebConversationFactory.serverAdapter= <code>com.ibm.vap.middleware.MiddlewareAdapter</code>	<code>com.ibm.vap.gateway.GatewayAdapter</code>	Selected adapter type : direct middleware or Gateway
PacWebConversationFactory.serverAdapter.location		The location used in the location file
PacWebConversationFactory.serverAdapter.locationsFile		Location file
PacWebConversationFactory.serverAdapter.host		Address of the machine on which the Gateway runs
PacWebConversationFactory.serverAdapter.port		Port number for the gateway
[CMVT]		
C CMVT string= <code>Create</code>		Label of the create transaction code
M CMVT string= <code>Modify</code>		Label of the modify transaction code
A CMVT string= <code>Cancel</code>		Label of the cancel transaction code
X CMVT string= <code>Create or Modify</code>		Label of the create/Modify transaction code.
[colors]		
Background= <code>White</code>		Color of the page background
Link color= <code>Blue</code>		Color of hypertext links
A Link color= <code>Red</code>		Color of activated links
Visited link color= <code>Red</code>		Color of visited links
[Screens]		
Common PageUp Key=		Label associated with the



- an `<ApplicationName>.ini` file which contains the application parameters.  
This file is used by the generator only, to find the generation parameters when the application is generated again.
  - a first page, required to initialize the Dialog, whose name is given by the `First HTML Page` field:  
`<FirstHTMLPage>.htm`
  - a page which informs that the user is no longer connected to the application: `EOTPAGE.htm`.
  - a page which displays the internal errors of the servlet: `servletexception.jsp`
  - an `<ApplicationName>.war` file in a J2EE Web Archive format to deploy the application
- Under the `app` sub-directory located under the `Generation_dir/Short_application_name` directory:
    - a utility page which contains JavaScript utility functions or the HTML screen descriptions in *frames* (=fields), in a file whose name is given in the `Short application name` field.  
`<ShortApplicationName>app.htm`
    - a page for each screen, stored in a file named `<screenCode>` (on 6 characters) and suffixed `htm`, `html` or `jsp` (HTML/JSP generation option):  
`<ScreenCode>.<suffix>`
    - the following files:  
`edittpl.htm`, `error.htm`, `fieldtpl.htm`, `hiddentpl.htm`,  
`labeltpl.htm`, `menubar.htm`, `notfoundtpl.htm`, `passwdtpl.htm`
- Under the `help` directory which is located in the `Generation_dir/Short_application_name` directory:
    - a help page named `<ShortApplicationName>help.htm`
    - a help page for each screen, contained in a file named `<ShortApplicationName>_help_<ScreenCode>` (on 6 characters) and suffixed `htm`.
    - an index page for each screen, contained in a file named `<ShortApplicationName>_index_<ScreenCode>` (on 6 characters) and suffixed `htm`.
    - `footer.htm`: icons for help screens
    - `helphelp`: help on help
    - `<ShortApplicationName>_Index`: general index
- Under the `js` directory which is located in the `Generation_dir/Short_application_name` directory:
    - a `public.js` file for public functions used at General level
    - a `private.js` file for private functions used at General level

- `<ShortApplicationName>private.js`: private functions used at Dialog level
- `<ShortApplicationName>public.js`: public functions used at Dialog level
- `<ScreenCode>private.js`: private functions used at Screen level
- `<ScreenCode>public.js`: public functions used at Screen level
- `style.css`: file used for the styles
- `notfoundtplprivate.js` and `notfoundtplpublic.js`

 For more information on the functions used at General, Dialog and Screen levels, refer to Chapter *Customizing generated pages*, Sub-chapter *JavaScript Functions*.

- under the `icon` directory which is located in the `Generation_dir/Short_application_name` directory:
  - the bitmaps used in the pages
- under the `WEB-INF` sub-directory of the `generation_dir` directory:
  - `web.xml`: file which describes the parameters of the web application and which can be used to modify the servlet behavior. The `traceLevel` parameter, set to `0` by default, can be changed to obtain traces (`0`: no trace, `4`: debug trace). This trace is redirected towards the standard output of the application server (terminal under WAS).
  - `lib` directory which contains the libraries required for the application (`vaprun.jar`, `vapweb.jar` or `vapwebctg.jar`).



When you re-generate a page, you lose all initial modifications entered in your HTML pages.

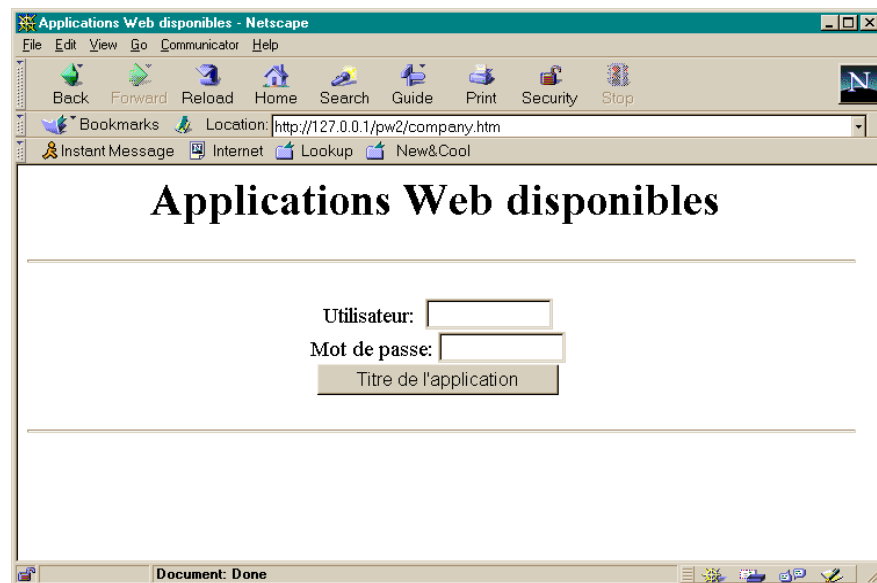


## Chapter 6: Presentation of Generated Pages

### First Page

The First page contains the User and Password fields used in an application whose server requires a user identification (CICS for example).

The label of the button used to start the application corresponds to the name you have entered in the **Application Title** field of the **Various parameters** page, in the generation options.

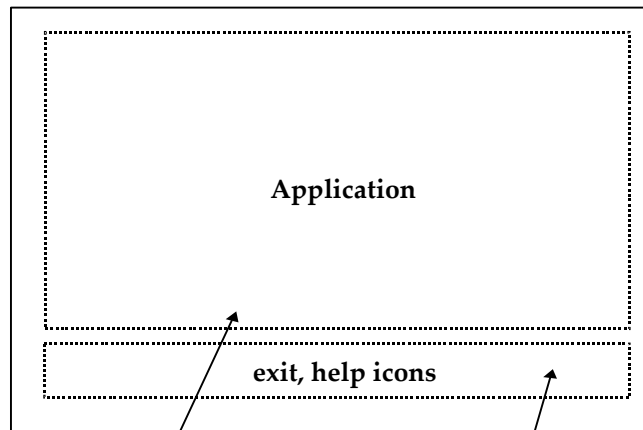


## Structure of the HTML Page

The HTML page is displayed in a general page which is made of two frames: the first one is used to display the application's screen and the other one to display the buttons which control the application window and the help window.

File name: `<ShortApplicationName>App.htm`

Function: defines two frames (`app` and `menubar`), loads JavaScript functions



Frame name: `app`

Filename contained in the frame: `<ScreenName>.htm`

Function: displays the requested screen

Frame name: `menubar`

Filename displayed in the frame: `menubar.htm`

Function: displays the control bar of the application



## Screen Description

The generated `<ScreenName>.htm` file contains the description of the screen.  
The screen is divided into two sections: the **HEAD** and the **BODY**.

<code>&lt;HTML&gt;</code>
<code>&lt;HEAD&gt;</code> Comments indicating that the screen results from a VisualAge Pacbase generation. META data which identifies the screen (program code, generation date, etc.) Screen title Call of the stylesheet Loading of JavaScript functions (at the screen level) <code>&lt;/HEAD&gt;</code>
<code>&lt;BODY&gt;</code> <code>&lt;FORM METHOD=POST&gt;</code> HIDDEN field used for the error message, the cursor positioning, the function key value Screen description: Literals INPUT and SELECT fields etc. <code>&lt;/FORM&gt;</code>
<code>&lt;/BODY&gt;</code> <code>&lt;/HTML&gt;</code>

### Head

This section contains the following data:

- identification of the program (META instructions),
- screen title,
- stylesheet name,
- loading of JavaScript functions.

*Example:*

<pre>&lt;TITLE&gt;screen title&lt;/TITLE&gt; &lt;META NAME=" SESSI" CONTENT=" 5222 "&gt; &lt;META NAME=" LIBRA " CONTENT=" xxx "&gt; &lt;META NAME=" DATGN " CONTENT=" 15012000 "&gt; &lt;META NAME=" TIMGN" CONTENT=" xxxxxxx "&gt; &lt;META NAME=" PROGE " CONTENT=" xxxxxx "&gt; &lt;META NAME=" COBASE " CONTENT=" xxxx "&gt; &lt;LINK REL=STYLESHEET HREF="/zz/app/style.css" type="text/css"&gt; &lt;SCRIPT LANGUAGE="JavaScript" SRC="/zz/js/zzxxxxPrivate.js"&gt; &lt;/SCRIPT&gt; &lt;SCRIPT LANGUAGE="JavaScript" SRC="/zz/js/zzxxxxPublic.js"&gt; &lt;/SCRIPT&gt;</pre>
---

## Body

This section contains the screen description. It contains a **FORM** instruction which specifies the '**POST**' action (option for the parameter transfer) and may include the list of controls to be performed (**onSubmit**).

The Body section has the following characteristics:

- Each new line creates a new line break (**<BR>** instruction) in a « Line » type generation.
- Each label or protected field is retrieved as a text, adapted to the generation parameters.
- Variable fields (protected or not):
  - **F**-type data element (displayed or protected on the screen but received by the program): a hidden field whose value is the label text must be added besides the label which is in a text form.
  - **P**-type data elements (displayed or protected on the screen but not received by the program): the data element is displayed as a simple text.
  - **V**-type data element (variable): each variable field is described with an **<INPUT>** or **<SELECT>** instruction, followed by the name of the corresponding data element (**NAME** attribute), the field length (**MAXLENGTH** attribute) and for input fields, the displayed length (equal to the maximum length at generation time, but it can be modified by the developer). This instruction does not contain the call to the appropriate control function; this is done in the submit (**onSubmit**-type Event Handler).

Example:

```
<INPUT TYPE='TEXT' NAME='COPOS' CONTENT='#COPOS000101'  
MAXLENGTH=5 LENGTH=5>
```

- The fields whose values are unknown (such as **TEXT** or **PASSWORD**) include a **CONTENT** (or **VALUE**) attribute whose value is the name of the field prefixed by **#** (see example above).
- Each field which has a default value is initialized with this value (**SELECTED** attribute of **BUTTON**-type **INPUT** fields, **SELECTED** instruction of **SELECT** fields, etc.).

## Structure of JSP Pages

JSP pages are constituted of a specific header which contains the import clauses, Java declarations and scriptlets, followed by an HTML part equivalent to that of HTML pages: only the generation of variable fields is different.

Variable fields are displayed via the call to a Java function (**DisplayField**) whose code can be seen in the **Private.jspf** file.

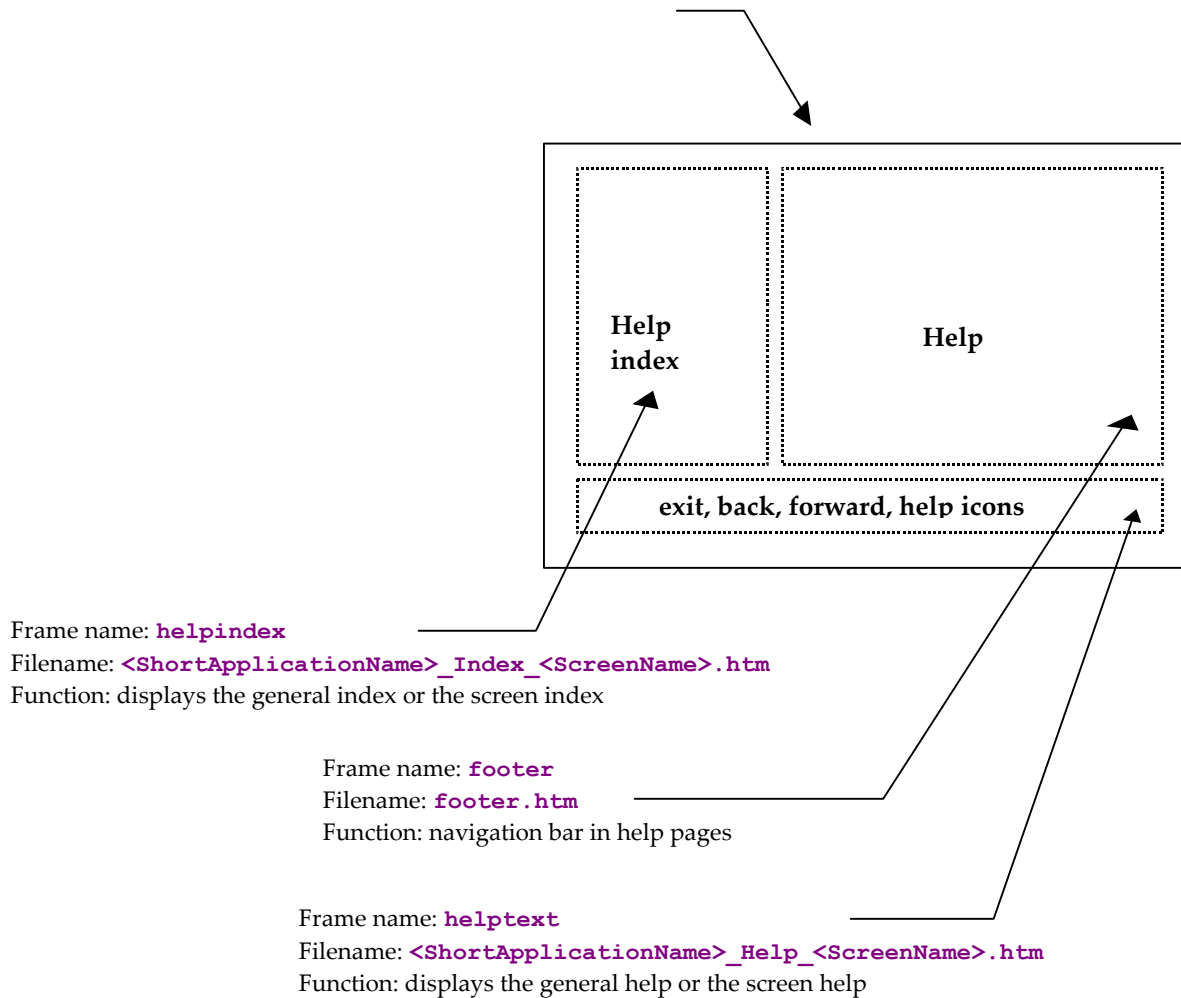
## Structure of Help Pages

Help pages are generated from the file extracted from VisualAge Pacbase.

Help is generated directly in HTML and saved on the HTTP server. Help is structured as follows:

Filename: `<ShortApplicationTitle>.help.htm`

Function: defines 3 frames `helpindex`, `helptext` and `footer`.



The `<ShortApplicationName>_Help.htm` file (the `Short application name` was given in the corresponding option of the `Required parameters` generation options) defines the size and location of the 3 frames. The 3 frames are described in 3 different files: (`footer.htm`, `<ShortApplicationName>_Index_<ScreenName>.htm` and `<ShortApplicationName>_Help_<ScreenName>.htm`)

## Contextual Help

A help page is generated for each screen. This page contains:

- the general help associated with the screen:  
`<ShortApplicationName>_Help_<ScreenName>.htm`  
It is displayed in the `helptext` frame.
- the list of the screen's data elements with their long label, authorized values and associated documentation.  
`<ShortApplicationName>_Index_<ScreenName>.htm`  
It is displayed in the `helpindex` frame.

## General Help (summary)

General help is displayed in the `helpindex` frame, i.e. the frame which can also contain the list of a screen's data elements.

General help contains the list of screens, each screen corresponding to a link on the help associated with this screen. The user then chooses the screen whose help he/she wants to view: the corresponding help page is displayed in the `helptext` frame and the screen index overwrites general help in the `helpindex` frame.

## The Navigation Bar

The navigation bar is displayed in the `footer` frame, and described in the `footer.htm` file. It contains the following icons:

- **Back** and **forward**: to view help pages already viewed. These icons call the `history.back()` and `history.forward()` Javascript functions which are located in standard Javascript functions.
- **Help** : to display general help in the `helpindex` frame.
- **Exit**: to quit help.

## Chapter 7: Customizing the Generated Templates

The generated HTML or JSP templates can be entirely customized. You may modify the aspect of a page (add colors, icons, modify labels, layout or insert fields) by using an HTML page editor or by modifying the HTML page directly via a text editor.



However for HTML templates, you must be careful not to delete the character strings which begin with #.

*Example: Customizing the First Page*

*The first page is always generated. By default it contains a title field and a button to start the application. But this can be modified. For example, you may add buttons to start various applications from the same first page. To do this, you must copy the following HTML paragraph, located in the description of the generated first page, into the file whose name you have entered in the **First HTML page name** field of the **Various parameters** generation option.*

```
<INPUT TYPE = "button" NAME = "WE" VALUE = "application Title"
onClick = "openNewWin('/WE/app/', 'WEApp')">
```

### Styles

The CSS1 (Cascading stylesheet) architecture is used for the presentation (size, font, colors...) of the various types of fields in a page. These styles are described in the **style.css** file under the generated **js** directory.

- There is one style for each type of field:
  - **text** for a text type field,
  - **passwd** for secret type field,
  - **libfix** for fixed labels,
  - **libvar** for variable labels,
  - **radio** for radio buttons,
  - **screenitle** for the title of a screen,
  - **dropdown** for drop-down lists.
- 3 styles when mapping repetitive fields:
  - **libtbl** for the table heading,
  - **rowodd** for odd rows,
  - **roweven** for even rows.
- and n styles are available to simulate the display of the Dialogs.

The names of these n styles are computed from the names of the presentation attributes in VisualAge Pacbase. There are 56 different styles named **.libXYC**, where **X** can take the **N** or **B** values (**Normal**, **Bold**), **Y** can take the **N**, **B**, **U**, **R** values (**Normal**, **Bold**, **Underline**, **Reverse**), and **C** can take **W**, **R**, **P**, **Y**, **G**, **T** and **B** values (**White**, **Red**, **Purple**, **Yellow**, **Green**, **Turquoise**, **Blue**).

Styles can be modified, they will not be overwritten when re-generating.

## Automatic Modification of HTML Pages

The generator can add comment tags upon generation (**Generate with comments** option in the **Various parameters** page). These tags make it easier to carry out automatic changes in the generated page. In this way, it is possible to apply the same modification to a set of pages or to apply the same changes again after a re-generation.

The generated tags are the following ones:

At the beginning of the screen line `<!--!PWC!LINEx-->` with `x`=Line number

Each field is framed either by

```
<!--!PWC!field name-->
```

or

```
<!--!PWC!xCy--> (for the labels)
```

*Example: Automatic deletion of a line and modification of the **SIZE** attribute in some of the fields.*

This example is written in Perl.

```
# print current line until arg1
sub printuntil
{
  unless (/${_}[0]/)
  {
    print OUT;
    while (<IN>) {
      last if (/${_}[0]/);
      print OUT;
    }
  }
}

# replace value of size if greater than 10
sub replacesize {
  &printuntil("SIZE=");
  s/SIZE=[0-9]{2,}/SIZE=15/;
}

# suppress a line
sub supline()
{
  local($wrk) = "<!--!PWC!LINE";
  local($search) = $wrk . $_[0] . "-->";

  if (/${search}/)
  {
    while (<IN>) {
      last if (/${wrk}/);
    }
  }
}
```

```

}

($filein) = @ARGV;
open (IN, $filein) || die "Cannot open file $filein
:$filein \n";
open (OUT, ">result.htm");
while(<IN>) {
    &supline("19");
    if (/<!--!PWC!NOMRUE/)
    {
        &replacesize();
        &printuntil("<!--!PWC!NOMRUE");
    }
    print OUT;
};

```

## JavaScript Functions

### Presentation of Javascript Functions

You can add controls or customize the aspect of each page by writing Javascript programs.

To manage HTML pages, the generated Javascript functions are divided into two categories: private and public functions.



Private functions are essential for the running of the whole application, they should not be modified by the user.

When generated, public functions are empty so that the user can enter his own processing. However their names should not be modified.

Unlike public functions, private functions are systematically regenerated when a generation is requested and all modifications are overwritten.

Furthermore, these functions are divided into 3 levels that we will name: General, Dialog and Screen: the General level functions are common to all dialogs, the Dialog functions are related to a particular Dialog and the Screen level functions are related to a particular screen.

If we take **WX** as the **short application name**, the names of the files which will store the functions will be:

	General	Dialog	Screen
<b>Public</b>	Public.js	WXPublic.js	xxxxxxPublic.js
<b>Private</b>	Private.js	WXPrivate.js	xxxxxxPrivate.js

### Functions used at General Level

#### *Private*

`openNewWin(strPath, strAppName)`

This function opens a new navigator window. It is used to display the help.

#### **Go (str)**

Function activated when a page is validated either by activating a key function or the « Enter » key

#### **ReadCook ()**

Reads the name and the password the user entered in the First Page.

#### **UpdatePosCursor (aObjectText)**

Memorizes the field where the entry point is located.

#### **SetPosCursor ()**

Sets the entry point.

#### **DisplayError ()**

Displays an error message in a JavaScript window if you asked this type of display at generation time.

#### **PWCLoad ()**

Function activated when a page is loaded.

#### **PWCUnload ()**

Function activated when a page is unloaded.

#### **PWCChange (objet)**

Function activated when a field is modified.

#### **PWCKeyPressed (e)**

This function is executed each time a key is pressed. It is used to submit a page by pressing the Enter key.

#### **PWCSubmit ()**

Function activated when a page is submitted.

#### Trace functions

The **TraceInit**, **Trace** and **TraceView** functions are used to add a trace.

**TraceInit** initializes a variable in which additional messages are stored by use of the **TraceView** function. The display of the data contained in this variable and thus the display of traces, is triggered by the call to **TraceView**.

These functions can be used in all public or private functions.

#### **Public**

#### **chkDate (type, field)**

This function is used for the date control of a field

#### **chkNum (field)**

This function is used for the numeric control of a field



#### **chkHere (field)**

This function is used for the presence control of a field

#### **displayErr (str, field)**

This function displays the errors detected by the control functions above.

#### **initArray (), isEmpty (str) , isANumber (str)**

Functions used to implement control functions.

### **Functions used at Dialog Level**

#### **Private**

It does not include any functions but a file is available for possible enhancements.

#### **Public**

All these functions are empty. They can be used as entry points to enter additional processing. They are called via General level functions.

#### **PWCLoadA ()**

Function called when an HTML page is loaded and **after** the specific processing defined at General level (Cf **PWCLoad ()**).

#### **PWCUnloadB ()**

Function called when an HTML page is unloaded and **before** the specific processing defined at General level (Cf. **PWCUnload ()**).

#### **PWCChangeA (object)**

Function called when a field is modified in an HTML page, **after** the specific processing defined at General level (Cf. **PWCChange (object)**).

#### **PWCChangeB (object)**

Function called when a field is modified in an HTML page, **before** the specific processing defined at General level (Cf. **PWCChange**).

#### **PWCKeyPressedB ()**

Function called when a key is pressed, **before** the specific processing defined at General level (Cf. **PWCKeyPressed (e)** ).

#### **PWCSubmitB ()**

Function called when an HTML page is submitted, **before** the specific processing defined at General level (Cf. **PWCSubmit ()**).

### **Functions used at Screen Level**

#### **Private**

#### **check ()**

This function is executed before submitting a page. It performs numeric, presence and date controls which may have been requested upon generation.

### **Public**

#### **PWCSCRLoadA ()**

Function called when the HTML page is loaded, **after** the specific processing defined at General level (Cf. **PWCUnload ()**) and **after** **PWCLoadA ()**.

#### **PWCSCRUnloadB ()**

Function called when an HTML page is unloaded and **before** the specific processing defined at General level (Cf. **PWCUnload ()**) and **before** the call to **PWCUnloadB**.

#### **PWCSCRChangeA (object)**

Function called when a field is changed in an HTML page and **after** the specific processing defined at General level (Cf. **PWCChange (objet)**) and after **PWCChangeA**.

#### **PWCSCRChangeB (object)**

Function called when a field is changed in a HTML page and **before** the specific processing defined at General level (Cf. **PWCChange (objet)**) and before **PWCChangeA**.

#### **PWCSCRKeyPressedB ()**

Function called when a key is pressed and **before** the specific processing defined at General level (Cf. **PWCKeyPressed (e)**) and before **PWCKeyPressedB**.

#### **PWCSCRSubmitB ()**

Function called when an HTML page is submitted and **before** the specific processing defined at General level (Cf. **PWCSubmitB ()**) and before **PWCSubmitB**.

## **Using JavaScript Functions as Entry Points**

Public JavaScript functions described above can be used to insert JavaScript code. This code is aimed at modifying the standard behavior of pages.

Depending on the level on which they apply (General, Dialog or Screen), these modifications will impact one or several pages.

*Example: Creation of a function which transforms lowercase values into uppercase values:*

*To systematically transform the lowercase values of the **NOVOL1000101** field into uppercase values in a given screen, you just need to change the code of the **PWCSCRChangeA** function defined in the **<Screen\_Name>Public.js** file.*

```
function PWCSCRChangeA(objet) {  
    var wfrom = objet.name;
```

```

        if (wfrom == "NOVOL1000101")
            objet.value = objet.value.toUpperCase()
    }

```

## Templates

### Presentation of Templates

The HTML template files generated by Dialog Web Revamping are built from template files supplied in Dialog Web Revamping.

These generation template files are located under the `htmltpl` and `jsptpl` directories (depending on the selected generation type (`HTML` or `JSP`)).

The HTML template files used upon generation contain labels which are dynamically replaced with the contents of the application files.

#### First Page

HTML page which contains the application starting button.

Template used	Template label	Replaced with the content of the following file
Firsthtmlpage.tpl	#firsthtmlpagetitle	firsthtmlpagetitle_xx.tpl
	#errorcompat	errorcompat_xx.tpl
	#user	User_xx.tpl
	#password	Password_xx.tpl
	#suffix	The suffix given at generation

xx corresponds to the selected language: fr for French and us for English.

#### Common Page

Page which contains the partition in frames and the local controls common to all screen pages.

Template used	Label	Replaced by
	#shortapp	Short application name given at generation
	#applicationtitle	Application name given at generation
	#servletcall	servletcall.tpl
	#menubar	Menubar.tpl
	#suffix	Suffix given at generation
	#msgnoframes	Msgnoframes_xx.tpl

xx corresponds to the selected language: fr for French and us for English.

## Screen Page

Page which contains the description specific to a screen. There are two types of templates depending on the format chosen for the page generation.

Template used	Label	Replaced by
Screen.tpl	#title	
	#shortapp	Short application name given at generation
	#screenname	VisualAge Pacbase screen name
	#body	Outfield.tpl Label.tpl Radio.tpl Editbox.tpl Hiddenfield.tpl Dropdownbox.tpl Table.tpl
	#enter	Submit button Submit_xx.tpl

## Help Page

Page which contains the partition in frames of help pages.

Template used	Tag	Replaced with
help.tpl	#msgnoframes	Msgnoframes_xx.tpl

Page which contains the help text

Template used	Tag	Replaced with
helpscreen.tpl	#label	helplabel_us
	#label	helplabel_fr
	#bodytag	bodytag.tpl
	#bg	backgroundfile.tpl

## Template Description

Templates	Description
Privatejspf.tpl	Private.jspf file which contains Java scriptlets
app.tpl	Common page which contains the partition in frames and the functions used for local controls
backgroundfile.tpl	Used to generate the filename which contains the background in the case of a file-type background
beginindex.tpl	Beginning of index pages
bodytag.tpl	Beginning of the BODY part of a page
bodytbl.tpl	Body of the generated page with option « Table »
bodytitle.tpl	Title of a screen page
Ccontrol.tpl	Used to create a cell when using a Table format
button.tpl	Submit button

Templates	Description
cell.tpl	Table cell
chkdate.tpl	Date control function
chkhere.tpl	Presence control function
chknum.tpl	Numericity control function
createwarfile.tpl	Contains the command which creates the .war file
dropdownbox.tpl	Drop-down list
editbox.tpl	Edit box
edittpl.tpl	Input field for a particular generation
endindex.tpl	End of index pages
eot_xx.tpl	End of transaction message
eotpage.tpl	Contains the 'End of Transaction' page
Epilibjarfile.tpl	List of the files copied when using CTG
errorcompat_fr.tpl	Error compatibility with the navigator in French
errorcompat_us.tpl	Error compatibility with the navigator in English
fieldtpl.tpl	Labelled field for a particular generation
firsthtmlpage.tpl	First page
firsthtmlpagetitle_fr.tpl	Title of first page in French
firsthtmlpagetitle_us.tpl	Title of first page in English
footer.tpl	help page footer
help.tpl	Partition of help pages in frames
helphelp_fr.tpl	French help on help
helphelp_us.tpl	English help on help
helpindex.tpl	Index page
helplabel_fr.tpl	French help title
helplabel_us.tpl	English help title
helplineindex.tpl	Link used by help indexes
helprub.tpl	Anchor for data element help
helpscreen.tpl	Page which contains all of a screen help
hiddenfield.tpl	Invisible field
hiddentpl.tpl	Template when generating particular hidden fields
indextitle_fr.tpl	French Title for index
indextitle_us.tpl	English Title for index
initparam.tpl	Parameter declaration in web.xml
itemradio.tpl	Generation of a radio button item
label.tpl	Label of a data element on a screen
labeltpl.tpl	Label used when generating 'on-the-fly'
libjarfile.tpl	List of the files copied when using the Middleware or the Gateway
lineindex.tpl	Link in the index of HTML help
menubar.tpl	Screen footer
Menubarpf.tpl	Screen footer with function keys.
msgdate_fr.tpl	Message for date control in French
msgdate_us.tpl	Message for date control in English
msgday_fr.tpl	Message for date control (day) in French
msgday_us.tpl	Message for date control (day) in English
msgfield_fr.tpl	Champ
msgfield_us.tpl	Field
msglongdate_fr.tpl	Long message for date controls in French
msglongdate_us.tpl	Long message for date controls in English
msgmonth_fr.tpl	Message for date controls (month) in French
msgmonth_us.tpl	Message for date controls (month) in English
msgnoframes_fr.tpl	Message to signal the use of frames in French
msgnoframes_us.tpl	Message to signal the use of frames in English
msgnumber_fr.tpl	Message for numericity controls in French
msgnumber_us.tpl	Message for numericity controls in English
msgvalue_fr.tpl	Message for a value request in French
msgvalue_us.tpl	Message for a value request in English

Templates	Description
notfound.tpl	Main template when generating 'on-the-fly'
OnClick.tpl	Contextual help for radio buttons
Onfocus.tpl	Contextual help for texts lists etc...
option.tpl	Choice for a drop-down list
outfield.tpl	Field
passwdtpl.tpl	Secret field when generating a particular page
password_fr.tpl	Password label for frontpage in French
password_us.tpl	Password label for frontpage in English
pfkey.tpl	Function key in menubarpf
private.tpl	Private JavaScript functions
privatedia.tpl	Private JavaScript functions at dialogue level
privatescr.tpl	Private JavaScript functions at screen level
public.tpl	Public JavaScript functions
publicdia.tpl	Public JavaScript functions at dialogue level
publicscr.tpl	Public JavaScript functions at screen level
radio.tpl	Radio button
row.tpl	Table row
screen.tpl	General skeleton for a screen
screenbutton.tpl	button
screenbuttonbar.tpl	Button bar when branching on screens by pressing buttons
servletcall.tpl	Servlet call
servletexception.tpl	Page displayed to signal an exception in the servlet
starteditor.tpl	Command which starts the location editor
style.tpl	Stylesheet
submit_fr.tpl	French label of submit button
submit_us.tpl	English label of submit button
table.tpl	Table
titleicon.tpl	Addition of a title icon
user_fr.tpl	User label for first screen in French
user_us.tpl	User label for first screen in English
Webxml.tpl	Generation of the web.xml file

## Modifying the Templates

Dialog Web Revamping Generator generates HTML pages from HTML template files.

The HTML template files used in the generation process contain HTML syntax blocks which can be modified. These blocks enable you to build an elementary paragraph.

It is possible to modify these templates so as to apply these modifications to all generated pages.

To modify a template, we advise you to duplicate, in the `template.usr` directory, the files located in the `template.htm` directory rather than modify these files directly.

## Chapter 8: Appendix

In this appendix, you will find all the information you need to modify an application generated with a version earlier than Dialog Web Revamping 3.5 to be able to use it with this 3.5 version

### Configuration of a Servlet Container

The **PacWebServlet** servlet must be configured in the Servlet Container via the **web.xml** file..

*Example of the **web.xml** file*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app id="APPN">
  <display-name>vols new</display-name>
  <servlet>
    <servlet-name>PacWebServlet</servlet-name>
    <servlet-class>com.ibm.vap.webconnection.html.servlet.PacWebServlet</servlet-
class>
    <init-param>
      <param-name>Suffix</param-name>
      <param-value>.htm</param-value>
    </init-param>
    <init-param>
      <param-name>Error message</param-name>
      <param-value>BOTH</param-value>
    </init-param>
    <init-param>
      <param-name>Instal_Dir</param-name>
      <param-value>/APPN/app</param-value>
    </init-param>
    <init-param>
      <param-name>traceLevel</param-name>
      <param-value>0</param-value>
    </init-param>
    <init-param>
      <param-name>PacWebConversationFactory.serverAdapter.codePageFile</param-
name>
      <param-value>D:\user\VapGateway\Charconv.txt</param-value>
    </init-param>
    <init-param>
```

```

        <param-name>PacWebConversationFactory.serverAdapter</param-name>
        <param-value>com.ibm.vap.middleware.MiddlewareAdapter</param-value>
    </init-param>
    <init-param>
        <param-name>PacWebConversationFactory.serverAdapter.location</param-
name>
        <param-value>PacWeb-LOCAL</param-value>
    </init-param>
    <init-param>
        <param-
name>PacWebConversationFactory.serverAdapter.locationsFile</param-name>
        <param-value>D:\user\VapGateway\vapLocat_pacweb.ini</param-value>
    </init-param>
    <init-param>
        <param-name>ConversationFactory</param-name>
        <param-
value>com.ibm.vap.webconnection.pwcm.PacWebConversationFactory</param-value>
    </init-param>

    <init-param>
        <param-name>endOfTransactionPage</param-name>
        <param-value>/EOTPAGE.htm</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>PacWebServlet</servlet-name>
    <url-pattern>/PacWebServlet</url-pattern>
</servlet-mapping>

<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>company.htm</welcome-file>
</welcome-file-list>
<error-page>
    <exception-type>javax.servlet.ServletException</exception-type>
    <location>/servletexception.jsp</location>
</error-page>
</web-app>

```



## Initialization Parameters of the PacWebServlet Servlet

These parameters are the following ones:

- **Suffix**: suffix used in the naming of HTML templates
- **Error message**: the display mode of error messages
- **Instal\_Dir**: the directory which contains the templates
- **TraceLevel**: trace level for the servlet **0**: no trace up to **4**: debug trace
- **PacWebConversationFactory.serverAdapter.codePageFile**: the transcoding file path
- **PacWebConversationFactory.serverAdapter**: the adapter selected for the middleware
- **PacWebConversationFactory.serverAdapter.location**: the location used in the location file
- **PacWebConversationFactory.serverAdapter.locationsFile**: the location file path
- **ConversationFactory** and **PacWebConversationFactory.serverAdapter**: used to instantiate the classes related to the selected middleware
- **EndOfTransactionPage**: page displayed when receiving an end of transaction
- **EPIConversationFactory.transactionId**: transaction code in CTG-mode
- **EPIConversationFactory.geoFile**: list of the extraction files used by the application
- **EPIConversationFactory.address**: address of the machine on which CTG runs
- **EPIConversationFactory.server**: server name for CTG
- **EPIConversationFactory.port** : port number of the CTG gateway
- **HtmlFiller**: this parameter makes it possible to inhibit the dynamic modification of the display of the fields. In this case you must add the following block:

```
<init-param>
  <param-name>HtmlFiller</param-name>
  <param-
value>com.ibm.vap.webconnection.PacWebHtmlFiller</param-
value>
```

```
</init-param>
```

## Migration of Applications (Versions 2.5, 3.0) towards J2EE Mode (Version 3.5)

Since Dialog Web Revamping version 3.5 is implemented by a servlet, a simple http server is no longer adapted : it is fundamental to use a servlet engine or a Web application server which supports servlets (e.g. Websphere Application Server, Tomcat...)

Let us take the example of an old application which is to be deployed using a **Root Context** `=/oldpacweb`:

### Calling the Servlet

The generated HTML pages must be modified in order to replace the call to the perl script with the call to the **PacWebServlet** servlet.

This modification must be done in 2 application files (`private.js` and `<AppCode>app.htm`) as well as in the generated first HTML page (ie `company.htm`)

In those 2 files, you must replace the call to the cgi script:

i.e. for `<AppCode>app.htm` :

```
wrk = "<frame src=" + '/cgi/perl.exe?cgicgi.pl%20' +
today.getTime() + '%20YOYO%20' + MyUser + '%20' + MyPasswd
+ '%20localhost' + "name='app' scrolling=yes>"
```

by the call to the servlet

```
wrk = "<frame src=" + '/pacweb/servlet/PacWebServlet' +
"name='app' scrolling=yes>"...
```

and for `private.js` the line

```
top.app.document.forms[0].action='/pacweb/pacweb/servlet/Pa
cWebServlet ...etc
```

is replaced with the line

```
top.app.document.forms[0].action='/pacweb/servlet/PacWebSer
vlet';
```



The call to the **ReadCook()** javascript function is no longer relevant and must be removed from the `<AppCode>app.htm` page.

## Managing the Cookies for the User/Password

In the authentication page (e.g. `company.html`), you must change the constitution of the cookie by naming it "`pacweb=`".

*Example :*

```
document.cookie="pacweb=@pwc@" + user + "@pwc@" + passwd;
```

## Adding the New Files

Under the application root (where the first page is), you must add the `EOTPAGE.<suffix>` and `servletexception.jsp` files. These pages can be found under the `htmltpl/template.htm/EOTPAGE.tpl` and `htmltpl/servletexception.tpl` directories. If you use the `EOTPAGE.tpl` template, you will have to replace the `#eot` string with the character string you want to display at the end of the transaction.

A new `WEB-INF` directory must be created. This directory contains the `web.xml` deployment file and the `lib` directory which contains the files used by the `PacWebServlet` servlet: `vaprun.jar` and `vapweb.jar`.

You must adapt the `PacWebConversationFactory.serverAdapter.locationsFile` and `PacWebConversationFactory.serverAdapter.location` parameters to your middleware configuration file (see below).

## Adding the RootContext in Pages

Each call to a resource, either a stylesheet or files which contain javascripts, must be modified so that the URL is prefixed by the Root Context which will be chosen to deploy the application.

*Example :*

```
LINK REL=STYLESHEET HREF="/APPN/js/style.css" type="text/css">
```

will be replaced with:

```
LINK REL=STYLESHEET HREF="/oldpacweb/APPN/js/style.css" type="text/css">
```

You can easily automate this modification by using a Perl script for example.

To easily deploy this application, you can create a `.war` file which will then be imported to the Web applications server.

*Example*

```
Jar -cf New.jar APPN WEB-INF servletexception.jsp EOTPAGE.htm
```

## VAP Middleware Configuration File

To use Java interfaces which make it possible to use a VAP middleware, you must create a file which describes the middleware configuration.

*Example :*

```

<Location-PacWeb-CICS>

MWARE= TCPMVS

MESSAGE_LENGTH=14200

MONITOR=WE1S

MWTIMEOUT= 2000

MWADDRESS= 9.134.0.0 9999

MWCODEPAGE= 297

MWTRANSID=WE1S

```

## Example of the Use of Conversation Proxies

You can use the conversation Proxy layer directly (i.e. independently of the `PacWebServlet` servlet). This layer is responsible for the communication towards the host application. You can then use the conversion Proxy layer as the source of data coming from the Dialog application.

```

public static void main(String[] args) {
    ConversationFactory conversationFactory;
    java.util.Properties properties = new java.util.Properties();
    // Initialisation pour utilisation GatewayAdapter
    properties.put(SERVER_ADAPTER, "com.ibm.vap.gateway.GatewayAdapter");
    properties.put(CONVERSATION_FACTORY,
        "com.ibm.vap.webconnection.pwcm.PacWebConversationFactory");
    properties.put("PacWebConversationFactory.serverAdapter.locationsFile",
        "D:/user/VapGateway/vapLocat_pacweb.ini");
    properties.put("PacWebConversationFactory.serverAdapter.location",
        "PacWeb-LOCAL");
    properties.put("PacWebConversationFactory.serverAdapter.codePageFile",
        "D:/user/VapGateway/CharConv.txt");
    properties.put("PacWebConversationFactory.serverAdapter.host",
        "127.0.0.1");
    properties.put("PacWebConversationFactory.serverAdapter.port", "8888");
    conversationFactory = new PacWebConversationFactory();
    conversationFactory.initialize(properties);

    Conversation conversation = conversationFactory.createConversation();

    int conversationState = -1;
    Screen newScreen = null;
    while(true)
    {
        conversationState = conversation.getState();
        try
        {
            switch (conversationState)
            {
                case Conversation.BEGIN :
                    { //lancement de la conversation, récupération du premier écran
                        newScreen = conversation.begin();
                        break;
                    }
                case Conversation.MIDDLE :
                    { //mise a jour de l'écran avec les nouvelles valeurs et poursuite
de la conversation
                        newScreen = conversation.nextScreen();
                    }
            }
        }
    }
}

```

```

        break;
    }
    case Conversation.END :
    { //fin de la conversation
        newScreen = conversation.end();
        break;
    }
}

if (newScreen != null)
{
    displayScreen(newScreen);
    if (newScreen.getCode().equalsIgnoreCase("WE00ME"))
    {

    }
    else
    {
        // Affiche les valeurs d'une iterative
        Field field;
        ListIterator ListedesPilotes =
newScreen.findUpdatableFieldsMatching(Screen.ITERATIVE_CATEGORY, "NOMPIL");
        while (ListedesPilotes.hasNext()) {
            field = (com.ibm.vap.webconnection.Field) ListedesPilotes.next();
            System.out.println("NOMPIL[" + field.getCategoryLineIndex() + "]= "
                + field.getStringValue());

            }
            // Mettre "FT" dans le code choix
            Field codeChx = newScreen.findUpdatableFieldMatching("COCHX");
            codeChx.setStringValue("FT");
        }
    }
} catch (ConversationException ce)
{
    System.out.println("ConversationException : " + ce.getMessage());
}
if (newScreen == null) break;
} //while
}

```

## Use with 'CICS Transaction Gateway'

This use is reserved to applications generated under CICS in native mode, i.e. where BMS maps are recognized. The application sends a physical message which corresponds to the BMS map ; this is the reason why the screen code is present in the message but the data related to some fields (field codes, category, nature...) is absent. This data is then searched for in the file output by the extraction of the file(s) which contain(s) the description of the VA Pac screens. (**GEO C4**) (As a consequence you must make sure that the file is in phase with the server part).

You must use Cics Transaction Gateway version 4.0 or greater to use this functionality.



The servlet engine must have been configured in order to have access to the **ctgclient.jar** file.