IBM® Workplace Forms™ Server — Webform Server

Version 2.6.1

Administration Manual

G325-2595-00

Note

Before using this information and the product it supports, read the information in “Notices,” on page 69.

First Edition (September 2006)

This edition applies to version 2.6.1 of IBM Workplace Forms Server — Webform Server (product number L-DSED-6RFRFZ) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Introduction

IBM® Workplace Forms™ Server — Webform Server translates XFDL documents into HTML/Java Script documents. This allows users to view, fill out, sign, and submit XFDL documents using only a web browser. In other words, users can fill out XFDL forms without downloading or installing browser plugins or other programs.

About This Document

This document explains how to install, configure, and maintain the components of Webform Server. For information about the differences between Webform Server and the Workplace Forms Viewer, and advice on designing forms that will work with Webform Server, refer to the *IBM Workplace Forms Server — Webform Server Best Practices Guide*.

Who Should Read This Document

This document is intended for system administrators responsible for installing, configuring and maintaining Webform Server components. This document assumes you are familiar with creating and deploying portlets and servlets.

Document Conventions

Placeholders

This manual uses "*xx*" or "*xxx*" in place of two or three digit version numbers when referring to file names, folders, and directories that contain version numbers.

Italics are used to represent the placeholders for specific paths, URLs, folder names and so on, and are enclosed by angle brackets (<>).

For example, the following placeholder denotes the directory in which you installed Webform Server:

```
<Webform Server Install Dir>
```

Choices

This manual presents choices by using the pipe symbol (|). This symbol means "or". For example, if a setting can be true or false, you may see:

```
true | false
```

System Requirements

Webform Server is a multi-platform server-side application. Hardware requirements will vary depending on the expected load and the complexity of your application. Furthermore, you may need to distribute or cluster your Webform Server installation.

For detailed information regarding Webform Server system requirements, see http://www.ibm.com/support/docview.wss?rs=2357&uid=swg27008290.

Backwards Compatibility

Webform Server supports forms in XFDL versions 6.0-6.5, as well as XFDL version7.0. However, if this version of Webform Server is used with a servlet or portlet application that was written for a previous version of Webform Server, that servlet or portlet must be updated to work with the latest Webform Server framework jar files. For more information about upgrading applications to work with newer versions of Webform Server, see “Upgrading Webform Server” on page 18.

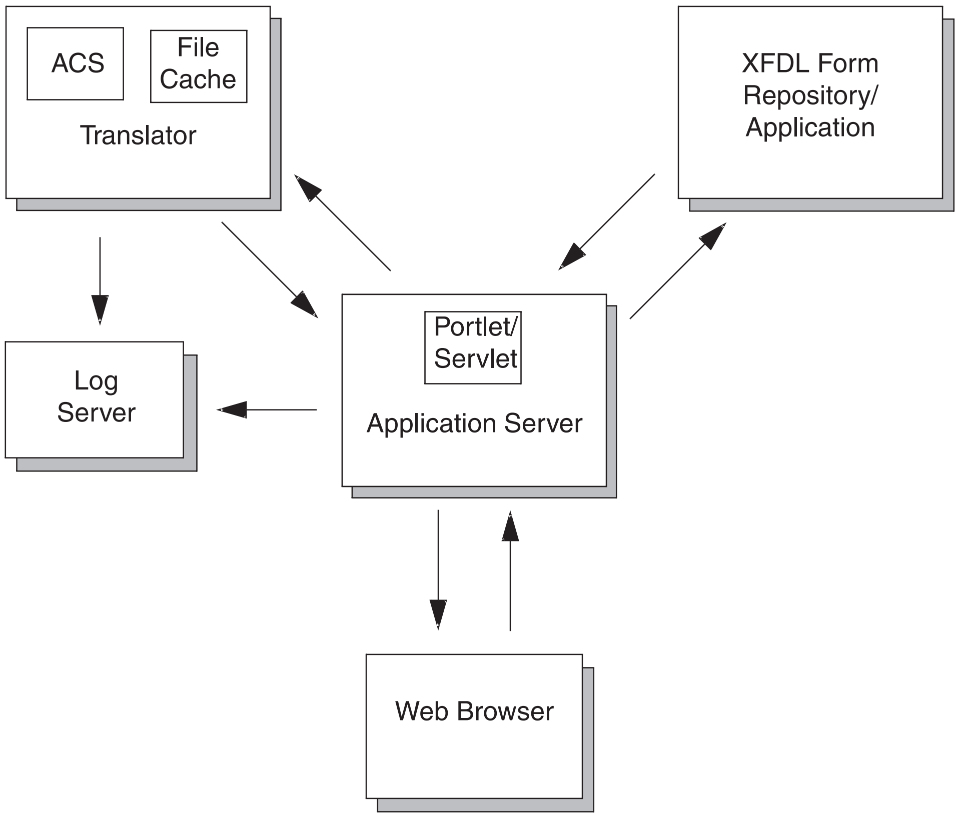
Webform Server Overview

Webform Server uses a collection of server components to intercept requests for XFDL forms, translate those forms into HTML and JavaScript™, and return the translated forms to the user. Webform Server can then receive completed HTML forms, translate them back into the original XFDL, and pass the completed XFDL forms to a processing application.

This allows end-users to view and fill the forms without installing any client software - instead, the user works with an enhanced HTML form in their regular web browser. However, because there is no client software, Webform Server does not offer all of the functionality available with Workplace Forms Viewer.

Basic Architecture

Webform Server relies on three central components: a portlet or servlet, a Translator, and a Log Server. The Translator also contains two sub-components: an Access Control Server (ACS) and a File Cache. The following diagram shows how these components are set up in a typical installation:



The portlet/servlet is installed on the application server. You can install the Translator and the Log Server on any computer that the application server can communicate with. Finally, you may also have a Form Repository or a Form Application from which the portlet or servlet retrieves XFDL forms.

The components perform the following functions:

- **Portlet/Servlet —** The portlet or servlet controls the forms application and processes all incoming and outgoing forms. It passes forms to the Translator for conversion, and may communicate with other applications or rely on a forms repository. This component is unique to each application, and must be written by you. Webform Server provides a Framework that includes methods for this purpose.
- **Translator —** The Translator converts forms between XFDL and HTML. When converting a form to HTML, the Translator stores the original XFDL in a file cache and saves some metadata about the form in access control list. When converting a form back to XFDL, the Translator retrieves the original form from the file cache and transfers the data from the submitted HTML form.
- **Log Server —** The Log Server logs activity for both the Translator and the portlet/servlet. The logs provide general transactional information, as well as error information.
- **XFDL Form Repository/Application —** This optional component may be a collection of XFDL forms that users may work with or an application that will return XFDL forms.

Modes of Operation

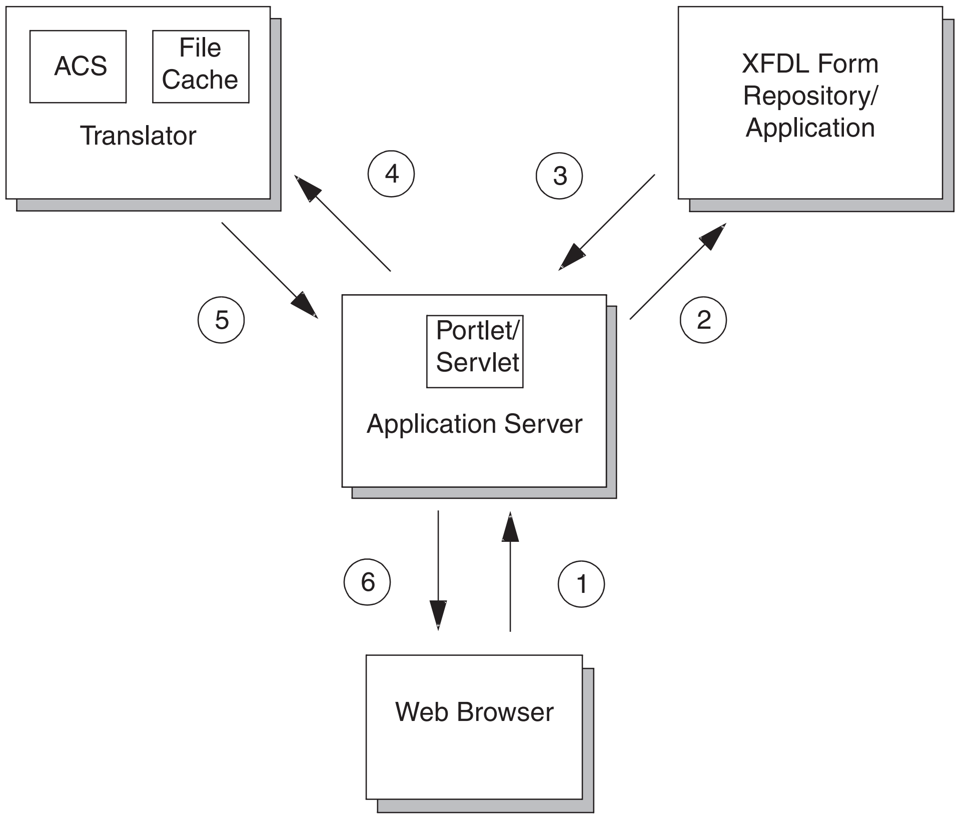
The Webform Server components work together to respond to the following actions:

1. **Form Requests —** When a user requests a form, Webform Server retrieves the requested XFDL form and translates the form into HTML and JavaScript before passing it back to the user.
2. **Form Submissions —** When a user submits a form, Webform Server translates the form back into XFDL, and then routes the form appropriately.
3. **Special Actions —** While working with a form, a user might request a special action, such as an update to all of the computes in the form. Webform Server receives these requests and responds to them with the appropriate information.

Each of these scenarios is further explained in the following sections. Note that the Log Server is excluded from these scenarios, since it simply logs information and does not affect the sequences that are described.

Requesting a Form

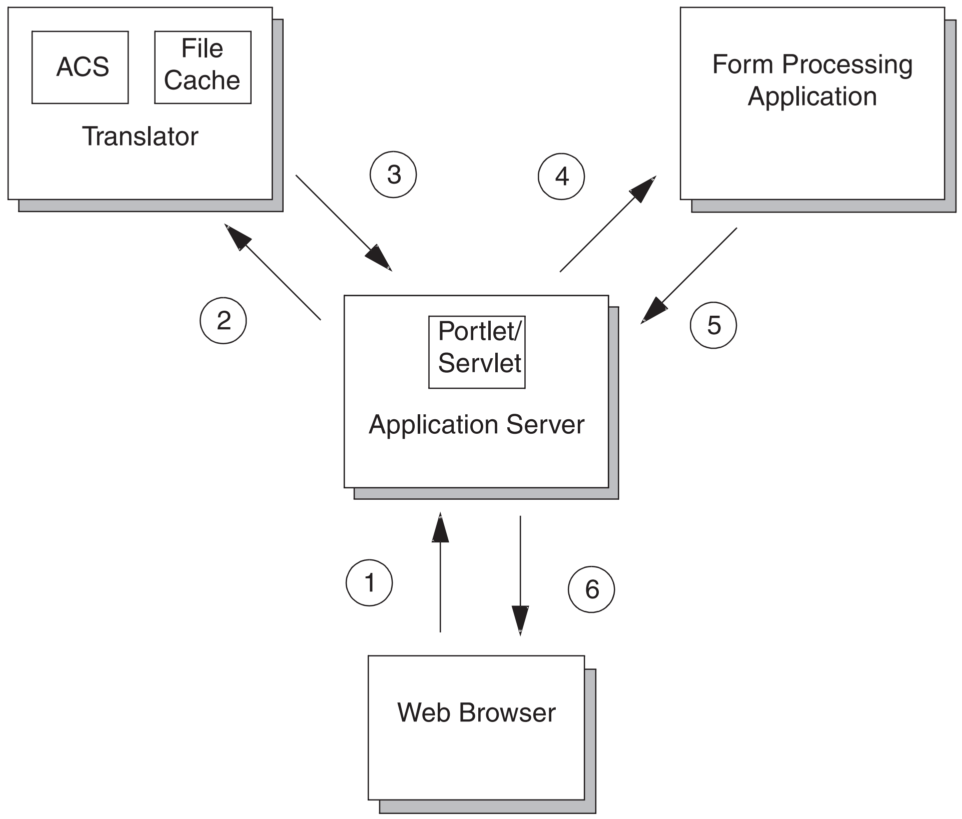
When the user request a form from the Application Server, Webform Server translates the form into HTML before sending it to the user. The following diagram shows the steps involved:



1. The user requests a form from the portlet/servlet. This request is usually triggered by clicking a link in a browser.
2. The portlet/servlet receives the request and locates the correct file in a Form Repository or triggers a Form Application which returns an XFDL form.
3. The form is passed to the portlet/servlet.
4. The servlet passes the form to the Translator.
5. The Translator converts the form to a combination of HTML and JavaScript and sends the converted form back to the portlet/servlet.
6. The portlet/servlet passes the converted form to the web server, which returns it to the user.

Submitting a Completed Form

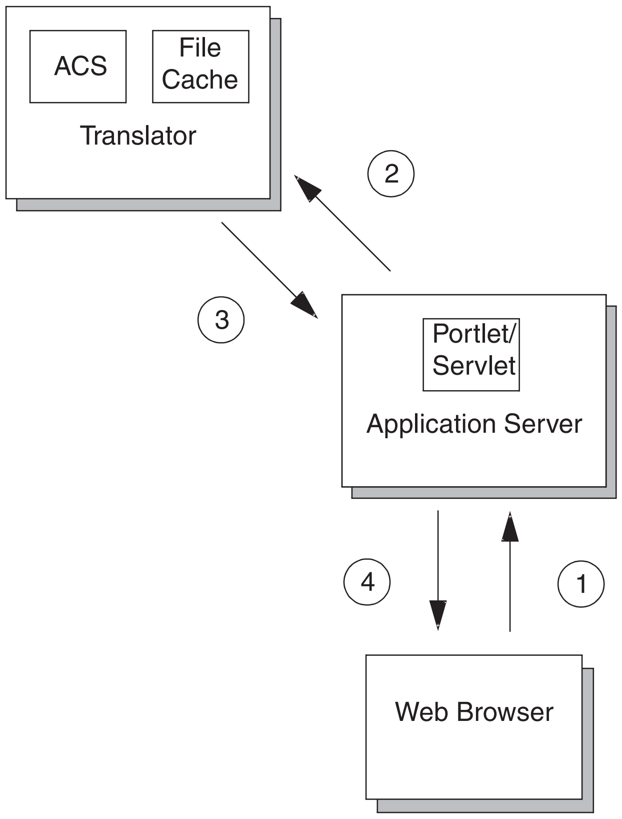
Once a form is completed, it is typically submitted for processing. Webform Server translates the form back into XFDL before passing it to the correct application, such as a workflow or a database, as shown:



1. The user submits the form to the portlet/servlet. Since the form is HTML, the submission consists of tag-value pairs.
2. The portlet/servlet receives the form and sends it to the Translator.
3. The Translator receives the submission, matches it to a previously cached XFDL form using certain metadata, and then maps the tag-value pairs of the HTML submission to the XFDL form. The Translator then passes the completed XFDL form to the portlet/servlet.
4. The portlet/servlet passes the XFDL form to the intended application.
5. The application generates a response and sends it back to the servlet/portlet.
6. The servlet/portlet forwards the response to the user.

Performing Special Actions

In some cases, users will make special requests to the server. These requests allow Webform Server to update the computes in an active form, display enclosures to the user, or save a copy of the XFDL form to the user's computer. These actions are performed through the portlet/servlet:



1. The user makes a special request, such as clicking a button to update the computes in the form or display an attachment. This request is sent to the portlet/servlet.
2. The portlet/servlet receives the request and sends it to the Translator.
3. The Translator interprets the request and takes the appropriate action. For instance, it may run an update on all of the computes in the form, or it may retrieve an attachment from the original XFDL form. The Translator then passes the information to the portlet/servlet.
4. The portlet/servlet sends the information to the user. For example, this may be an updated form or a Word document that was attached to the form.

Differences Between Webform Server and Workplace Forms Viewer

When using Webform Server, users do not need to install any client-side software. Instead, they can rely on their standard web browser for viewing and filling forms. However, this also limits the functionality available to users, since some functionality cannot be provided without client software.

For a complete discussion of the differences between Webform Server and Workplace Forms Viewer, refer to the *IBM Workplace Forms Server — Webform Server Best Practices Guide*.

Note: You do not need to be familiar with these differences to install and run Webform Server, but you should review these differences if you are designing forms for use with Webform Server.

About Installing Webform Server

You can install Webform Server on a single computer, as a distributed system, or as a clustered system. The following sections detail those installations, and provide information to help you decide which installation is most suitable.

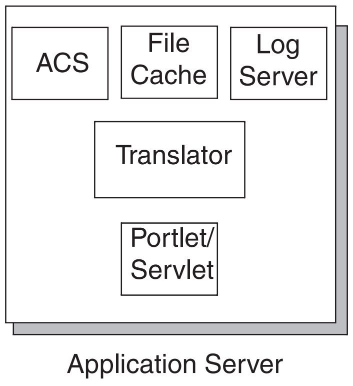
Choosing How to Install Webform Server

Before installing Webform Server, you must decide whether you want to install it on a single server, or whether you want to install it in a distributed environment. Webform Server operates normally in either configuration. However, installing it in a distributed environment may be preferable in the following scenarios:

- If your web server is hosting a large number of applications and performance is a concern.
- If you have extremely high traffic and need to load balance your system.

About Installing on a Single Server

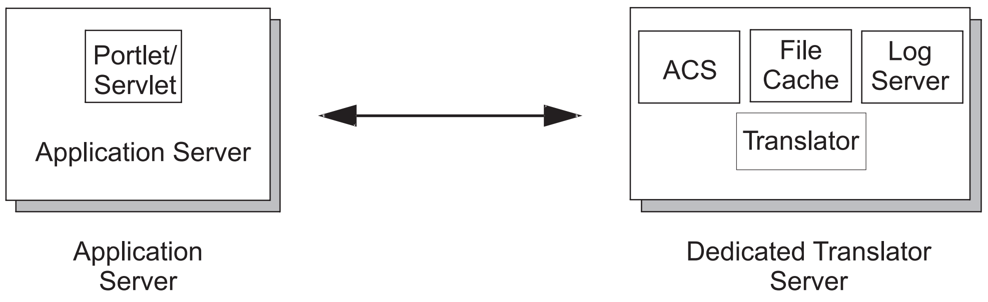
In this configuration, you install all components of Webform Server on a single computer, as shown:



Due to performance considerations, this configuration is recommended only for development or testing purposes. For production purposes, you should run Webform Server in a distributed environment.

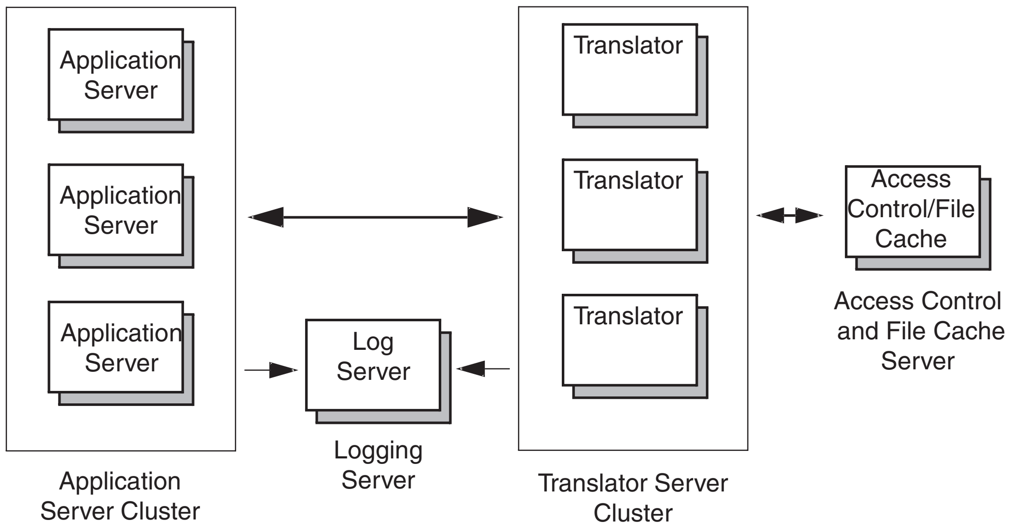
About Installing in a Distributed Environment

Installing Webform Server in a distributed environment is preferable for production deployments. In this scenario, you should move the Translator and related components to a dedicated server, as shown:



About Installing in a Clustered Environment

If you are processing a particularly heavy volume of forms, you may want to replicate your servers and implement load balancing. For example, you might have multiple Application Servers or Translator Servers, as shown:


Application Server
Application Server
Application Server
Application Server Cluster
Log Server
Logging Server
Translator
Translator
Translator
Translator Server Cluster
Access Control/File Cache
Access Control and File Cache Server

There are a number of software packages that will enable clustering. For more information about how to set up a clustered environment, refer to the instructions that accompany your clustering software.

Serving Both HTML and XFDL Forms

By default, Webform Server will automatically detect whether the client has the Viewer installed, and will serve either XFDL or HTML forms as appropriate. Additionally, you can create servlets or portlets that set specific response types.

For more information about implementing a servlet or portlet, refer to ″Creating a Servlet″ or ″Creating a Portlet″.

Installing Webform Server

This section provides instructions for installing Webform Server on a single server, and in a distributed or clustered environment. For more information about setting up a cluster, refer to the documentation that came with your clustering software/hardware.

Note: If you are upgrading from a previous version of Webform Server, refer to “Upgrading Webform Server” on page 18.

Before Installing Webform Server

Before installing Webform Server, you should consider the following factors:
- Placement of the shared file cache
- Port availability
- Partition Size

The items are discussed in more detail in the following sections.

Placement of the Shared File Cache

For best performance, you should consider the following when choosing the location of your file cache:

- Use a fast file server.
- If you are installing the shared file cache as part of a distributed environment, install it on the same server as the Translator.
- If you are setting up a clustered environment, we recommend that you install the Access Control Server and the shared file cache on the same server.
- Do not store non-Webform Server files in the shared file cache, as it is aggressively cleaned.

Additionally, when setting up the shared directory, you should limit read/write access. This will improve the overall security of your Webform Server installation.

Port Availability

By default, Webform Server uses the following ports:

- **8085 —** Webform Server uses this as a communication port in WAS.
- **4560 —** The Log Server uses this port to communicate with the Translator.
- **44445 —** The Access Control Server uses this port to communicate with the Translator.

If these ports are not available on your server, you must either free the ports or reconfigure the appropriate software. For port 8085, this means reconfiguring the default port in the WAS Administrative Console (see the documentation for WAS for details). For ports 4650 and 44445, you must reconfigure Webform Server (see "Configuring Webform Server" on page 21).

Partition Size

Before installing Webform Server, ensure that the partition that will contain the shared file cache has 20 GB or more of free space. This space is required to store the temporary files that Webform Server creates when users work with forms. The amount of space required varies, but is based on the following factors:

- Larger forms will require more space, as the temporary files will also be larger.
- More concurrent users will require more space, as more temporary files will be created.
- Longer times for completing forms will require more space, as temporary files will be on disk for longer.
- Less frequent cache cleaning will require more space, as temporary files will be on disk for longer.

If you feel that you are using particularly large forms, such as forms with many images, or that you have a particularly large user base, you should ensure you have plenty of hard drive space.

Installing Webform Server On a Single Server

To install Webform Server on a single server, follow these steps:

1. Reboot the server immediately before installing Webform Server.
2. Ensure that you are logged in as an administrator or root user.
3. Run one of the following installation programs:
   - **Windows**®: WFServer_*xxx*_WebformServer_Win32.exe
   - **Linux**®: WFServer_*xxx*_WebformServer_Linux.bin

- **Solaris**: WFServer_*xxx*_WebformServer_Solaris.bin
- **AIX®**: WFServer_*xxx*_WebformServer_AIX.bin

4.
5. Next, select the language you want to use.
   - The Webform Server Installer begins.
6. On the **Welcome** page, click **Next**.
   - The license agreement appears.
7. Read the terms of the license agreement, and click **Accept** to accept it.
   - You must agree to the terms of the licence agreement to continue the installation.
8. Next, the installer will ask you to choose the directory in which to install Webform Server. Do one of the following:
   - To accept the default folder, click **Next**.
   - To set a different folder, click **Browse**, locate and select the folder you want, click **OK** to confirm your selection, then click **Next** to continue.
     – The default install locations are:

       Windows: C:\Program Files\IBM\Workplace Forms\Server\x.x\\ Webform Server

       UNIX®: opt/IBM/Workplace_Forms/Server/*x.x*/Webform_Server
9. A new dialog box appears, displaying the following Webform Server components:
   - Shared File Cache
   - Access Control Server
   - Log Server
   - Translator
   - Webform Server Samples
     – Servlet Sample
     – Portlet Sample

   Ensure that all components are selected and click **Next**.
10. The following dialog box allows you to indicate whether you are deploying the Translator component to WebSphere® Application Server. Do one of the following:
    - To automatically deploy to WebSphere Application Server, select **Deploy to WebSphere Application Server** and click **Next**.
    - To automatically deploy both the Translator component and the sample servlet to WAS, select **Deploy to WebSphere Application Server** and **Deploy Sample Servlet**.
    - To deploy with an alternate servlet runner or to manually configure WebSphere Application Server, clear both options and click **Next**.

    Note: If you choose not to automatically deploy to WebSphere Application Server, the install still includes the Translator files. However, your existing WebSphere Application Server configuration is not changed.
11. If you chose to deploy the Translator to WebSphere Application Server, go to step 9. Otherwise, skip to step 10.
12. The next dialog box requires you to specify where Webform Application Server is installed. Do one of the following:
    - To accept the default folder, click **Next**.

- To set a different folder, click **Browse**, locate and select the folder you want, click **OK** to confirm your selection, then click **Next** to continue.

The **Translator Deployment Details** dialog box appears.

13. Under **Profile**, ensure that the installer has displayed the correct profile name for your WebSphere Application Server configuration. This option is only available for WebSphere Application Server 6.x and greater.
14. Under **Cell**, ensure that the installer has displayed the correct cell name for your WebSphere Application Server configuration.
15. Under **Node**, ensure that the installer has displayed the correct node name for your WebSphere Application Server configuration.
16. Under **Server Name**, type the name you want to give to the Translator server.
17. Under **Application Name**, type the name you want to give to the Translator application, then click **Next** to continue.
    - If you chose to deploy the Sample Application, the **Sample Servlet Deployment Details** dialog box appears. Otherwise, the **Installation Summary** dialog box appears.
18. If the Sample Servlet Deployment Details dialog box appears, go to step 16. Otherwise, skip to step 19.
19. Under **Application Name**, type the name of the Sample Servlet application. This can be any name, as long as it is valid under WebSphere Application Server.
    - The default application name is: **WebformSampleApp**.
20. Under **Application Context**, type the context you want to give to the Sample Servlet application.
    - The default context is: **/Samples**.
21. Click **Next**.
    - The **Installation Summary** dialog box appears.
22. Click **Install** to start the installation.

The installer will install all Webform Server components in the folder you selected. Once you have installed Webform Server, we recommend that you test your installation. See “Testing the Installation” on page 16 for more information.

Installing Webform Server in a Distributed or Clustered Environment

These instructions apply to both distributed and clustered environments, but do not provide information about clustering the components once they are installed. For more information about clustering, refer to the documentation that came with your clustering software.

When installing Webform Server in a distributed or clustered environment, you must install the shared file cache first.

Installing the Shared File Cache

1. Locate the server that will run the shared file cache.
2. Reboot the server immediately before beginning installation.
3. Ensure that you are logged in as an administrator or root user.
4. Run one of the following installation programs:
   - **Windows**: WFServer_*xxx*_WebformServer_Win32.exe

- **Linux**: WFServer_*xxx*_WebformServer_Linux.bin
- **Solaris**: WFServer_*xxx*_WebformServer_Solaris.bin
- **AIX**: WFServer_*xxx*_WebformServer_AIX.bin

   The Webform Server Installer begins.
5. Read the terms of the license agreement, and click **Yes** to accept it.
   - You must agree to the terms of the licence agreement to continue the installation.
6. Next, the installer will ask you to choose the directory in which to install Webform Server. Do one of the following:
   - To accept the default folder, click **Next**.
   - To set a different folder, click **Browse**, locate and select the folder you want, click **OK** to confirm your selection, then click **Next** to continue.
     - The default install locations are:

       Windows: C:\Program Files\IBM\Workplace Forms\Server\x.x\Webform Server

       UNIX: opt/IBM/Workplace_Forms/Server/*x.x*/Webform_Server
7. A new dialog box appears, displaying the following Webform Server components:
   - Shared File Cache
   - Access Control Server
   - Log Server
   - Translator
   - Samples
     - Servlet Sample
     - Portlet Sample

   Ensure that only **Shared File Cache** is selected and click **Next**.
8. The installer will ask you to specify a location for Webform Server.
   - The installer will use this as a root location, and will create a sub-directory called SharedFileCache to contain the file cache.
   - The installer will notify you when the installation is complete.
9. Once the installation is complete, locate the SharedFileCache directory and do the following:
   - Share the directory so that your Translator servers can access it.
   - Set appropriate permissions for the directory. For example, you may want to limit read/write access.

Installing the Other Components

You must run the installer on each computer that will host a Webform Server component. The installer allows you select which component you will install each time you run it.

To install a component:
1. Determine which component you want to install, and locate the appropriate server.
2. Reboot the server immediately before beginning installation.
3. Run one of the following installation programs:
   - **Windows**: WFServer_*xxx*_WebformServer_Win32.exe

- **Linux**: WFServer_*xxx*_WebformServer_Linux.bin
- **Solaris**: WFServer_*xxx*_WebformServer_Solaris.bin
- **AIX**: WFServer_*xxx*_WebformServer_AIX.bin

   The Webform Server Installer begins.
4. Read the terms of the license agreement, and click **Yes** to accept it.
   - You must agree to the terms of the licence agreement to continue the installation.
5. Next, the installer will ask you to choose the directory in which to install Webform Server. Do one of the following:
   - To accept the default folder, click **Next**.
   - To set a different folder, click **Browse**, locate and select the folder you want, click **OK** to confirm your selection, then click **Next** to continue.
     - The default install locations are:

       Windows: C:\Program Files\IBM\Workplace Forms\Server\x.x\Webform Server

       UNIX: opt/IBM/Workplace_Forms/Server/*x.x*/Webform_Server
6. A new dialog box appears, displaying the following Webform Server components:
   - Shared File Cache
   - Access Control Server
   - Log Server
   - Translator
   - Samples
     - Servlet Sample
     - Portlet Sample

   Choose the component you want to install and click **Next**.
7. The installer will ask you to choose the directory where you want to install the component. Do one of the following:
   - To accept the default folder, click **Next**.
   - To set a different folder, click **Browse**, locate and select the folder you want, click **OK** to confirm your selection, then click **Next** to continue.

The installer will install the selected Webform Server components. Once you have installed Webform Server, we recommend that you test your installation. See "Testing the Installation" on page 16 for more information.

Linux: Installing Fonts

Before you can use Webform Server on a Linux server, you should also install the fonts that are used by your forms. Webform Server uses these fonts both to create the HTML that is sent to the user and to generate an image of the form when the user prints. Note that Webform Server only uses TrueType fonts under Linux.

The default installation of Webform Server checks the following directories for font information (in the order listed):

```
<PWS_home_account>/Translator/fonts/
<PWS_home_account>/java/jre/lib/fonts/
```

If you have not installed the font Webform Server requires, it will instead locate the default font (Lucida Sans Regular) in the Java™ fonts directory and use it. However, be aware that relying on the default Java font may cause layout problems in your forms which will be especially noticeable when printed.

With this in mind, we recommend that you install your own fonts. To do this, you may have to purchase the fonts required. Once you have the fonts, you can simply copy the font files into the following folder:

```
<PWS_home_account>/Translator/fonts/
```

Note: You can change both which folders are searched and which font is used by default by changing the fontDirectory and defaultFontName settings in the translator.properties file. For more information, see "Translator Properties" on page 22.

Testing the Installation

Once you have installed Webform Server, you should test your installation to be sure it is working correctly. Before you do this, ensure that you have a supported web browser installed on your Translator server.

To test the installation, log on the computer that is running the Translator and open the welcome form:

- On Windows, click **Start** → **Programs** → **Webform Server** → **Webform Server Welcome Page**.
- On Linux, open a browser and go to the following URL:

```
<server_name>:8085/translator/Translate?pwsAction=toolbelt
```

Your web browser should display the welcome form.

Once you have tested Webform Server and are certain it is operating correctly, you should review "Configuring Webform Server" on page 21 to determine whether you need to change any of the default settings. Additionally, you may want to run one of the samples. For more information, refer to "Working with the Samples" on page 49.

Starting and Stopping Webform Server Processes

When working with Webform Server, you may find that you need to stop and restart its processes.

Windows

To stop and restart the processes on Windows:

1. Open the Administrative Tools console.
2. Right-click the following processes and select **Start** or **Stop**:
3.

Translator Service
IBM WebSphere Application Server V5 - IBMWebformServerTranslator26

Access Controller
IBM Workplace Forms Server - Webform Server Access Controller

Log Server
IBM Workplace Forms Server - Webform Server Logger

UNIX

To stop and restart the processes on a UNIX server:

Translator Service
Go to /opt/IBM/Workplace_Forms/Server/*x.x*/Webform_Server/Translator

Type `./stopTranslator.sh` to stop the process.

Type `./startTranslator.sh` to restart the process.

Access Controller
Go to /opt/IBM/Workplace_Forms/Server/*x.x*/Webform_Server/ACServer

Type `./stopACServer.sh` to stop the process.

Type `./startACServer.sh` to restart the process.

Log Server
Go to /opt/IBM/Workplace_Forms/Server/*x.x*/Webform_Server/LogServer

Type `./stopLogServer.sh` to stop the process.

Type `./startLogServer.sh` to restart the process.

Troubleshooting the Installation

If Webform Server is not working properly after installation, check the following:

- Ensure that ports 8085, 4560, and 44445 are free.
- If you are running Webform Server on Windows, ensure that the following services are operating correctly:
  - IBM WebSphere Application V*x* IBMWebformServerTranslator*xx* (Translator in WAS)
  - Workplace Forms Webform Access Controller (Access Control Server)
  - Workplace Forms Webform Logger (Log Server)
- If you are running Webform Server on Linux, ensure that the following java processes are running:
  - Access Control Server (com.ibm.form.webform.accesscontrol.server.AccessControlServer)
  - Log Server (com.ibm.form.webform.logging.log4j.server.LogServer)
  - Run the following command to determine the Translator Server's status:
    ```
    <WAS_Install_Dir>/AppServer/bin/serverStatus.sh TranslatorServer
    ```

For information about specific error messages you may encounter, refer to "Troubleshooting" on page 63.

Upgrading Webform Server

Before you upgrade Webform Server, make sure that you have a backup copy of any portlet or servlet you created to work with Webform Server. This can be your original development copy, or a copy you take from your current installation.

To upgrade Webform Server:

1. Uninstall all Webform Server components (see "Uninstalling Webform Server" on page 18).
2. Install the new version of Webform Server (see "Installing Webform Server" on page 10).
3. Update any portlet or servlet that you built to work with Webform Server so that they are compiled and packaged with the ws_framework.jar and ws_common.jar files from the new version (see "Packaging the Servlet" on page 36 or "Packaging the Portlet" on page 46).
4. Ensure that all the old PureEdge jar files are deleted to avoid class loader conflicts in the servlet containers.
5. In your web.xml file, update the servlet-class to: `com.ibm.form.webform.framework.portlet.WebformPortletForwarder`
6. In your portlet code, ensure that you change all references to PureEdgePortlet to IBMWorkplaceFormsServerPortlet, including the import statement.
7. Deploy the updated portlet or servlet (see "Deploying the Servlet" on page 36 or "Deploying the Portlet" on page 47).

Uninstalling Webform Server

Windows

When uninstalling Webform Server from Windows servers, you should follow these steps:

- Locate the Webform Server components
- Locate the shared file cache
- Uninstall the components
- Verify the uninstall

Locating the Webform Server Components

Because different components may be on different servers, you should begin by identifying which components are running on your server.

1. Open the Windows Services monitor. For example, on Windows 2003 Server you would open:

   Start → Settings → Control Panel → Administrative Tools → Services
2. Look for the following entries in the Services list:
   - **IBM WebSphere Application Server V*x* IBMWebformServerTranslator*x.x* —** This means the Translator is running on the server.
   - **Workplace Forms Webform Access Controller —** This means Access Control Server is running on the server.
   - **Workplace Forms Webform Logger —** This means the Log Server is running on the server.

Locating the Shared File Cache

The shared file cache will not appear as a Windows service. In general, you can expect it to be on the same server as the Access Control Server. To find the exact location of the shared file cache:

1. On a server with the Translator installed, open the translator.properties file.
2. Check the *fileCacheLocation* setting. This will list the path to the shared file cache.

For more information about the translator.properties file, see ″Translator Properties″.

Uninstalling the Components

On each server that is running one of the Webform Server components, run the uninstall process as follows:

1. Open the Windows **Control Panel** and select **Add/Remove Programs**.
2. In the **Add/Remove Programs** window, select **Webform Server**.
3. Click **Change/Remove**.
   - When the uninstall process prompts you to reboot your computer, do so.

Verifying the Uninstall

Once you have uninstalled one or more Webform Server components, you should verify that the Webform Server directory has been removed. If it has not been removed, manually delete it.

UNIX

When uninstalling Webform Server from UNIX servers, you should follow these steps:

- Locate shared file cache
- Locate the Webform Server components
- Uninstall the components
- Remove the automatic start commands
- Verify the uninstall

Locating the Shared File Cache

Before removing the other Webform Server components, you should locate your shared file cache. To do this:

Open the translator.properties file on one of your Translator Servers, and check the *fileCacheLocation* setting. For more information about the translator.properties file, see ″Translator Properties″.

If you are not sure which server is running the Translator, refer to the next step to locate the correct server.

Locating the Webform Server Components

Because different components may be on different servers, you should begin by identifying which components are running on your server:

1. Check the running java processes, as shown:
   ```
   ps -A uwww|grep java
   ```

2. Look for the following processes:
   - **Access Control Server**: com.ibm.form.webform.accesscontrol.server.AccessControlServer
   - **Log Server**: com.ibm.form.webform.logging.log4j.server.LogServer
   - **Translator**: Run the following command to determine the Translator Server's status: `<WAS_Install_Dir>/AppServer/bin/serverStatus.sh TranslatorServer`

Uninstalling the Components

On each server that is running one of the Webform Server components, run the uninstall process as follows:

1. Log in as root.
2. Run: `Webform_Server/uninst/uninstaller.bin`.
   - Some log files may remain after Webform Server has been uninstalled. You can safely delete these files.

Configuring Webform Server

Once you have installed Webform Server, you may need to change the default configuration to reflect your system. This section describes the configuration settings for the following components:

- WebSphere Application Server
- Access Control Server
- Translator
- Log Server

Working with the Access Control Server

Webform Server uses an Access Control Server to maintain an access control table that tracks form instances. The AC table associates a globally unique identifier (GUID) for each form instance with user information that the Translator provides. The AC table also tracks whether a form instance is in use, and the last time a form instance was accessed.

In this way, the AC table tells the Webform Server which forms are in process and allows the Translator to handle multiple forms from multiple users. Moreover, the AC table maintains security for Webform Server, ensuring that documents are delivered and updated only by the client session originating the document requests.

Cache Cleaning

The Access Control Server runs a thread to delete form instances that are no longer in use. This removes information about the instance from the Access Control Server and deletes the supporting files from the shared file cache. You set the frequency of this cleaning in the Access Control Server properties file.

Access Control Server Properties

The Access Control Server is configured through a server.properties file. This file is located in:

```
<Webform Server Install Dir>\ACServer\
```

The properties file is a standard text file which you can open in a text editor and change as necessary. However, if you make changes to this file while Access Control Server is running, those changes will not take affect until the Access Control Server is restarted.

The following table describes the settings available in the properties file. Note that the property names are case sensitive.

Property	Mandatory	Description
port	Yes	An integer. Identifies which port the Access Control Server should use.

Property	Mandatory	Description

fileCacheLocation	Yes	The path to the shared file cache. Use one of the following formats:
		• Network share:
		`\\<fileserver>/<filecache>`
		• Local Windows directory:
		`<drive>/<path>`
		• Local Linux directory:
		`/<path>`

retriesOnMutex	Yes	An integer. The number of times the Access Control Server will attempt to lock a form instance for use. The Server may be unable to lock the form if it is already in use.
hibernationTime	Yes	An integer. The amount of time, in milliseconds, that the Access Control Server will wait between attempts to lock a form. Default: **500** milliseconds.
cacheExpiry	Yes	An integer. The period of time, in seconds, from the last successful contact with a form instance until the form is considered inactive. Inactive forms are cleaned from the cache. Note that this time should never be less than the *contactFrequency* setting the Translator's translator.properties file. If it is, Webform Server may remove active form instances from the cache. We recommend that you set the *cacheExpiry* to be at least three times the *contactFrequency*. Default: **2700** seconds.
cleanerInterval	Yes	An integer. The frequency, in seconds, of the cache cleaning. Default: **180** seconds.

Working with the Translator

This section provides details about working with the Translator, which is a Java servlet running under WAS. The Translator converts forms between XFDL and HTML/Java Script.

The Translator uses the Shared File Cache to temporarily store information about the original XFDL document, such as enclosures and images. It also accesses the Access Control Server to track form instances and GUIDs.

Translator Properties

Each Translator is configured through a translator.properties file. This file is located in:

```
<Webform Server Install Dir>\Translator\TranslatorApp.ear\
```

The properties file is a standard text file which you can open in a text editor and change as necessary. However, if you make changes to this file while Translator is running, those changes will not take affect until the Translator service is restarted.

The following table describes the settings available in the properties file. Note that the property names are case sensitive.

Property	Mandatory	Description

fileCacheLocation	Yes	The path to the shared file cache. Use one of the following formats:
		• Network share:
		`\\<fileserver>/<filecache>`
		• Local Windows directory:
		`<drive>/<path>`
		• Local Linux directory:
		`/<path>`

fcCacheSize	No	An integer. The maximum number of forms to cache in memory. Default: **30**.
fcTimerDelay	No	An integer representing a number of seconds. Indicates how frequently the form cache is cleaned. Default: **10**.
changeNotification Items	No	Indicates which types of form changes made by the user are posted to the server. Valid settings are: • **none** — No user changes are posted to the server. Note that with **none** set, some AJAX calls are still passed to the server. • **format** — Only user changes to items with formatting are posted. • **all** — All user changes are posted. Default: **all**.
focusNotification Items	No	Indicates whether the user's cursor position is posted to the server. Valid settings are: • **none** — Position is not posted • **all** — All focus changes are posted. Default: **all**.
changeNotification Mode	No	Determines how the Translator component posts change notifications to the server. Valid settings are: • **synch** — Posts notifications synchronously. This option prevents users from typing in the next field while processing takes place on the server. • **asynch** — Posts notifications asynchronously. This option allows users to type into the next field while processing takes place on the server. Default: **asynch**.

Property	Mandatory	Description
autoRefresh	No	Indicates whether the form should auto-refresh if there is a change in the form that cannot be shown in real-time. Valid settings are: • **true** — Allows auto-refresh. • **false** — Does not allow auto-refresh. Default: **true**.
logServerHostname	No	Name of the host running the Log Server. For example, ”mylogserver.xyz.com”. Default: **localhost**, which assumes that the Log Server runs on the same server as the Translator.
logServerPort	No	An integer. The port that the Log Server uses to receive log info from the Translator. Default: **4560**.
acHostName	No	Name of the host running the Access Control Server. For example, “controlserver.xyz.com”. Default: **localhost**, which assumes that the Access Control Server runs on the same server as the Translator.
acPort	Yes	An integer. The port that the Access Control Server is using for requests. Default: **44445**.
acConnectRetries	No	An integer. The number of times the Translator will attempt to connect to the Access Control Server. Default: **5**.
contactFrequency	No	An integer representing seconds. Sets how often a form instance (that is, a form an end user is completing) contacts Webform Server to indicate that the form is still active. This contact is then passed on to the Access Control Server, which maintains the state of the form and cleans inactive forms from the shared file cache. Default: **900** seconds.
maxPrintProcesses	No	An integer. Sets the number of concurrent print processes allowed by the server. Allowing too many print processes at once can cause server performance to degrade. This value can range from 0 - 50. Note that if you set this to zero, the print button is removed from the toolbar when the user views the form. Default: **3**.

Property	Mandatory	Description
printPageWidth Inches	No	Allows you to set the printable width for a page. The printable area is dictated by the end-user's browser. Internet Explorer uses 3/4″ margins when printing. For a standard 8.5″ x 11″ page, this mean your printable width is 7″. If you do not set the width correctly, the browser will scale the image when printing, which may result in loss of detail. Default: **7**.
printPageHeight Inches	No	Allows you to set the printable height for a page. The printable area is dictated by the end-user's browser. Internet Explorer uses 3/4″ margins when printing. For a standard 8.5″ x 11″ page, this mean your printable height is 9.5″. If you do not set the height correctly, the browser will scale the image when printing, which may result in loss of detail. Default: **9.5**.
printScrollbarsOn Fields	No	Set to **true** to print scrollbars on fields. Set to **false** to omit scrollbars when printing fields. Note that fields that have no scrollbars in the Viewer will never print with scrollbars. Default: **true**.
printClassic Checkboxes	No	Set to **true** to print check boxes with check marks when selected. Set to **false** to print check boxes with an X when selected. Default: **false**.
printRadioToCheck	No	Set to **true** to make radio buttons look like check boxes when printed. Set to **false** to print radio buttons as radio buttons. Default: **true**.
printRadiosEmpty	No	Set to **true** to print all radio buttons as blank (unselected). Set to **false** to print radio buttons as they appear, with one selected. Default: **false**.
tabCharacter	No	A character. Tabs in labels are automatically replaced by a series of characters to ensure labels are sized properly for the tab. This sets which character is used. Default is the space character. Note that this only affects printing.
tabCharacter Count	No	An integer. Tabs in labels are automatically replaced by a series of characters to ensure labels are sized properly for the tab. This sets how many characters are used to replace the tab. Default is **8**. Note that this only affects printing.
printDPI	No	The DPI to use when generating the image for printing. Default is **300**.

Property	Mandatory	Description
printGrayscale	No	Set to **true** to generate the print image as a grayscale GIF rather than a color PNG. This will reduce the overall size of the print image, and will improve overall performance. Default is **false**.
autoClosePrint Preview	No	If **true**, the print preview will automatically close after the user prints the form. If **false**, the print preview will remain on screen after the user prints the form, allowing the user to save the image or print again. Default: **true**.
detectBrowser	No	Sets the response if Webform Server detects an unsupported browser on the client side. The following settings are valid: **ignore** — ignores the problem and serves the form to the user. **warn** — warns the user about the problem, then serves them the form. **halt** — displays an error message to the user, and does not serve the form. Default is **warn**.
fontDirectory	Yes	A comma separated list of directories that contain font files. When looking for a font, Webform Server will search the directories in the order listed, and will use the first match found.
defaultFontName	Yes	The name of a particular font. Webform Server will substitute this font when it cannot locate the font specified by the form. If Webform Server cannot locate the default font, the Translator service will not start.
validFileExtensions	No	A comma-separated list of file extensions that should open from the local file system. Default: **.xfdl, .xfd, .frm**.
showWizardButton	No	Determines whether the Document Accessibility Wizard is available. The following settings are valid: **true** — displays the Document Accessibility Wizard button. **false** — removes the button from the toolbar. Default: **false**.
extensionsDirectory	No	Specifies the directory that contains Webform Server extension files (ifx files).

Note: The **printPNGCompressionLevel** property is no longer supported. Webform Server now automatically uses the maximum level of compression when sending the print image to the user.

Setting Up SSL

To ensure that your clients' data is protected, you should configure Webform Server to use SSL.

In general, there are two ways to configure SSL:

- Enable SSL within WAS
- Configuring WAS to work with an external HTTP server which uses SSL

In either scenario, you should refer to your WAS documentation for configuration details.

Working with the Log Server

The Log Server is a central logging service that records activity for the Translator and any servlet or portlet written for Webform Server. The Log Server creates different logs for each application, and offers a number of configuration options.

About the Log Files

The Log Server stores the log files for each application (the Translator, a servlet, or a portlet) in different directories. These directories are named for the application, as shown:

```
Translator: <Webform Server Install Dir>/LogServer/Logs/Translator
Servlet:    <Webform Server Install Dir>/LogServer/Logs/Servlet
Portlet:    <Webform Server Install Dir>/LogServer/Logs/Portlet
```

Within each directory, the Log Server creates two different logs:

- **Operational Log —** This log records normal form events, such as form creation, form destruction, transactions involving enclosures, and so on.
- **Error Log —** This log records error events, such as exceptions or access control list failures.

Data Logged

The logs are written either as XML files or comma delimited text files, depending on how the Log Server is configured. They use the following tags to identify the data:

- **userInfo —** The IP address of the user.
- **formInstance —** A unique identifier for the form. This is the same identifier that is used to store the form in the shared file cache.
- **enclosureAction —** The type of action requested for an enclosure. This may be one of extract, display, remove, or attach.
- **enclosureName —** The filename of the enclosure. This is the same name that is used to store the enclosure in the shared file cache.
- **message —** The log message. The content of this message will vary depending on the context in which it is used.
- **errorHistory —** The complete stack trace for the error.
- **current_ACLInfo —** The IP address of the user that has locked that instance of the form.

The following list describes the events that are logged:

New Form Requested (Operational)
Logs the following information when a form instance is created:

- userInfo
- formInstance

Enclosure (Operational)
Logs the following information when an enclosure-related action occurs:

- enclosureAction
- enclosureName
- userInfo
- formInstance

Exception Thrown (Error)
Logs the following information when the Translator cannot find the Access Control Server or Shared File Cache:

- message
- errorHistory
- formInstance

ACL Check Failure (Error)
Logs the following information when the Translator cannot verify a form's GUID or IP against the access control table:

- current_ACInfo
- formInstance

Configuring the Log Server

The Log Server is configured through the logserver.properties file, which is located in the following directory:

```
<Webform Server Install Dir>\LogServer\
```

The properties file is a plain text file that you can edit in any standard text editor. The file is divided into the following sections:

- **General Settings** — This section contains some general settings that apply to all logs.
- **Translator Settings** — This section controls how the Log Server handles the logs for the Translator.
- **Servlet Settings** — This section controls how the Log Server handles the logs for servlets.
- **Portlet Settings** — This section controls how the Log Server handles the logs for portlets.

Each setting is written as a standard tag-value pair. Individual settings are also identified as applying to either the Translator or the Servlet, and are further identified as applying to either the operational log or the error log, as shown:

```
<application>_<log type>.<setting> = <value>
```

For example, to create a setting for the Translator's error log, you would use the following syntax:

```
translator_error.<setting> = <value>
```

And a setting for a servlet's operational log would use this syntax:

```
servlet_operational.<setting> = <value>
```

The following table describes the properties you can set for each log:

Property	Mandatory	Description
port	Yes	The port number to use for the Log Server. This setting is the same for all logs, and does not use the *<application>_<log type>* identification.

Property	Mandatory	Description
xsltPath	Yes	Full path to the Log Server's XSLT file directory. This setting is the same for all logs, and does not use the *<application>*_*<log type>* identification.
enabled	No	Set to **true** to enable logging for this category, or **false** to disable logging. Default is **false**.
ntEventLog	No	Windows only. Set to **true** to log events to the Windows Event Log for this category, or **false** to disable logging. If **true**, the **NTEventLogAppender.dll** file from *<logserver_path>*\ bin\ must be copied to the system folder of the server running the Log Server. Default is **false**.
console	No	Set to **true** to log events for this category to the STDERR, or **false** to disable logging. Default is **false**.
fileRollBySize	No	Set to **true** to include events for this category in the backup file when the log is rolled over, or **false** to disable this. Default is **false**.
fileRollBySize. fileName	No, unless *<operational* or *error>*.file RollBySize is true	Full path and filename of the XML backup file to which category events are to be logged. This property is required if ***<operational*** or ***error>*.fileRollBySize** is **true**.
fileRollBySize. maxFileSize	No	An integer that sets maximum size of the log file. Once the log file reaches this size, it is rolled into a backup file and a new file is created. The integer is followed by one of the following suffixes: • KB (kilobytes) • MB (megabytes) • GB (gigabytes) Default is **10MB**.
fileRollBySize. maxLogfiles	No	An integer that sets maximum number of backup log files kept. When this number is exceeded, the Log Server deletes the oldest backup log file. Default is **10**.
fileRollBySize. append	No	Set to **true** to have the Log Server append information to the existing log file at start up, or **false** to have the Log Server start a new log file at startup (in which case the existing log file is rolled over). Default is **true**.
fileRollBySize. logToCSV	No	Set to **true** to use comma separated values in the logs, or **false** to use XML. Default is **false**.

Sample Properties File

The following is an example of a logserver.properties file that sets operational values for both the Translator logs and the servlet logs:

```
port=4560
xsltPath=c:\\WebformServer\\LogServer\\xslt
translator_operational.enabled=true
translator_operational.fileRollBySize=true
translator_operational.fileRollBySize.fileName=c:\\WebformServer\\
   LogServer\\logs\\translator\\operational.xml
translator_operational.fileRollBySize.append=true
translator_operational.console=false
servlet_operational.enabled=true
servlet_operational.fileRollBySize=true
servlet_operational.fileRollBySize.fileName=c:\\WebformServer\\
   LogServer\\logs\\servlet\\operational.xml
servlet_operational.fileRollBySize.append=true
servlet_operational.console=false
```

In the above example, the first section sets the path to the XSLT files. This path is used when creating all logs, and does not require any further identification.

The second section sets values for the Translator's operational log. In this case, operational logging is enabled and writes an XML log. The logs are set to roll over when they reach 10 MB in size, and a total of 10 separate log files will be maintained, with older log files being deleted.

Finally, the third section sets the values for the servlet's operational log to reflect those of the Translator's operational log.

Maximizing Performance

As of version 2.6, Webform Server offers a more dynamic user experience, similar to the one offered by Workplace Forms Viewer. Essentially, Webform Server now automatically updates some computed elements or performs automatic refreshes instead of requiring manual refreshes by users.

To do this, Webform Server caches a number of forms in memory. While those forms are in memory, the dynamic form items refresh automatically with little delay. However, once the limit of the cache is reached, Webform Server begins to write the older forms to disk, which will slow server performance and response times on those forms.

The number of forms that Webform Server holds in memory is configurable. However, to maximize the performance of your application, you must determine the size of the form cache in memory. The default size is **30** forms. This presumes the following configuration:
- Single instance of Webform Server
- 4 GB of RAM
- Minimum heap size for the JVM process set to 256 MB (-Xmx)
- Maximum heap size for the JVM process set to 512 MB (-Xmx)
- Average form size of 50 MB (once loaded into RAM)

The following calculations determine the default size of the form cache:

Windows Servers
1. Under a 32-bit architecture, Windows can address up to 4 GB of virtual memory. 2 GB of that memory is reserved by the Windows operating system, with the remaining 2 GB available for the Webform Server process.

2. Of the 2 GB of memory available to the Webform Server process, 512 MB is reserved for the JVM (-Xmx), leaving 1.5 GB of native memory available for the form cache and other native libraries.
3. 1.5 GB of memory, divided by an average form size of 50 MB (after being loaded into RAM), allows 30 forms per server.

Smaller forms, with fewer pages, fewer items per page, less logic, and fewer computes may have a smaller average form size in memory. By the same token, larger, more complicated forms will have a larger form size. Be sure to take the size of your forms (after being loaded into RAM) into consideration when calculating how many forms you can maintain in the form cache.

UNIX Servers

As UNIX servers allocate less virtual memory to the operating system, you can increase the size of the form cache to **50** forms, as shown by the following calculation:

1. With 4 GB of virtual memory, 1 GB of virtual memory is reserved by the UNIX operating system, leaving 3 GB available for the Webform Server process.
2. Of the 3 GB of memory available to Webform Server, 512 MB is reserved for JVM (-Xmx), leaving 2.5 GB of native memory available for the form cache and other native libraries.
3. 2.5 GB of memory, divided by an average form size of 50 MB allows 50 forms per server.

Note: Do not set **fcCacheSize** to 0. This is not a recommended setting. A form cache of 0 will prevent any forms from being stored in memory and will force them to be stored on disk. This results in slower response times when tabbing through formatted fields and executing computes.

Changing the Java Heap Size

The Java heap size is determines how much memory should be reserved for the JVM process. By default, Webform Server sets both the initial and the maximum Java heap size (-Xms and -Xmx settings) to **512 MB**. This should be sufficient for most Webform Server deployments.

However, if your solution requires more memory to be allocated to the Java heap, you can modify these settings through the WebSphere Application Server Administration Console. You should change both the -Xmx and -Xms settings to the same value to prevent the JVM from dynamically allocating and de-allocating memory at run time.

Note: If you change the amount of memory allocated to the JVM, you must recalculate the number of forms that you can maintain in the form cache and update the **fcCacheSize** variable accordingly.

Changing the Size of the Form Cache

If you are using a UNIX server or your configuration differs from the one used to calculate the default size, you will need to change the size of your form cache.

To do this:

1. Calculate the size of your form cache, using the formula described in “Maximizing Performance” on page 30.

2. Open the translator.properties file.
3. Search for the **fcCacheSize** variable and update its setting.
4. Save the file.

Cleaning and Synchronizing the Form Cache

The **fcTimerDelay** variable determines how frequently the form cache is cleaned and synchronized with the disk cache. The default time is **10** seconds. You should maintain this default time unless you are running Webform Server on a clustered system. For clustered environments, this value should be lowered to ensure that the form cache and disk cache remain synchronized.

There is no recommended frequency for cleaning and synchronizing the form cache in a clustered environment, as each system will have its own requirements. You should test your system to determine the frequency that works best with your particular deployment.

To change the file cache size:

1. Determine how frequently your form cache needs to be cleaned and synchronized.
2. Open the translator.properties file.
3. Search for the **fcTimerDelay** variable and update its setting.
4. Save the file.

Creating a Servlet

If you want to implement your application as a servlet, you must create a special servlet that communicates with Webform Server. This servlet represents your overall application, and routes forms between the user and the Translator when necessary.

The servlet can include any functionality you want. For example, you might create a servlet that not only uses the Translator to provide HTML forms to the user, but that also routes completed forms to the next step in their life cycle.

To create this servlet, you must extend the IBMWorkplaceFormsServerServlet class that is provided with Webform Server. This class also provides a number of methods that will simplify the process of implementing your servlet.

Normal Servlet Operations

Normal operation of Webform Server assumes that there are two form-related actions the user might take:

1. The user requests a form from the servlet. This is generally a standard GET action that is triggered by clicking a link on a web page.
2. The user submits a completed form to the servlet through a POST action. This may be either an XFDL form or a form that was previously translated into HTML.

In either case, the request is processed through a combination of functionality that is provided by the IBMWorkplaceFormsServerServlet class and code that you write while extending the servlet. Furthermore, the IBMWorkplaceFormsServerServlet class is designed to ignore requests that do not involve XFDL forms. This means that all other operations will fall through the Framework to your custom code.

In general, you must implement a *doGet* and *doPost* method within your servlet to handle the expected form-related scenarios.

The doGet Algorithm

In general, the *doGet* method must respond to a request for an XFDL form. This presumes that you have created a link somewhere, perhaps on a web page, that triggers the servlet and uses one or more parameters to indicate which form is being requested.

Given this scenario, the following algorithm presents a simple way to respond to such a request:

1. User clicks a link that requests an XFDL form from the servlet.
2. The request falls through the IBMWorkplaceFormsServerServlet to your *doGet* method.
3. In your *doGet* method, you retrieve one or more parameters from the request object.
4. Based on those parameters, you load a form from the local file system, write the form to the output stream of the response object, and set the content type of the response object to **application/vnd.xfdl**.

- Once you have written the form to the response object, you can end your *doGet* method. The IBMWorkplaceFormsServerServlet object will continue operations from that point.

5. The IBMWorkplaceFormsServerServlet automatically detects whether the user has the Viewer, and returns the form in either XFDL or HTML as appropriate.

The doPost Algorithm

In general, the *doPost* method must respond to the submission of a completed form. This may be either an XFDL form or a previously translated HTML form. In either case, the following algorithm presents a simple way to respond to such a request:

1. User submits the form to the Servlet.
2. If the form was previously translated into HTML, the IBMWorkplaceFormsServerServlet automatically translates the form back to XFDL, then passes it through to your *doPost* method. If the form is XFDL, it is automatically passed through to your *doPost* method.
3. In your *doPost* method, you process the form as required. This may include saving the form to disk, archiving the form to a database, or sending the form to the next stage of a workflow.
4. If a response is required, write the response to the output stream of the response object and set the content type appropriately. Otherwise, you can end your *doPost* method.
   - For example, you might send an HTML page to confirm receipt of the submission, or you might load a second form from disk and return it to the user.
5. If a response is set, the IBMWorkplaceFormsServerServlet will send it to the user. If the response is an XFDL form, the IBMWorkplaceFormsServerServlet will automatically detect whether the user has the Viewer, and will translate the form into HTML if necessary. If no response is set, operation will cease.

Implementing a Servlet

You must implement your servlet by extending the IBMWorkplaceFormsServerServlet class, which is provided as part of Webform Server. The IBMWorkplaceFormsServerServlet class is itself an extension of the standard HTTP servlet class defined in javax.servlet.http.HttpServlet. As such, it inherits the methods defined for an HTTP servlet, and you are free to use or overwrite those methods.

Note: Do not overwrite the service method. This method is used exclusively by the IBMWorkplaceFormsServerServlet.

In addition, the IBMWorkplaceFormsServerServlet class provides a collection of custom methods to assist you in writing your servlet. These methods provide form-related functionality, such as setting the response type to XFDL or HTML, determining whether the client has the Viewer installed, and so on.

For more information about the IBMWorkplaceFormsServerServlet class and its methods, refer to the Webform Server Javadocs on the IBM website.

A sample servlet is also provided with Webform Server, including the source code used to implement the servlet. For more information, refer to “Working with the Samples” on page 49.

Setting Up Your System to Use the IBMWorkplaceFormsServerServlet Class

Before you can use the IBMWorkplaceFormsServerServlet class, you must add the ws_framework.jar, ws_common.jar, and log4j.jar to your classpath. These files are installed with Webform Server in the following location:

```
<Webform Server Install Dir>\redist\
```

Using Other Form API Methods During Development

You may want your servlet to access XFDL forms directly. For example, you may want to prepopulate forms as you load them, or extract data from forms when they are submitted. To do this, you must install the Workplace Forms Server — API and use it in conjunction with the IBMWorkplaceFormsServerServlet class to develop your servlet.

For more information about the API, refer to the *IBM Workplace Forms Server — API Installation & Setup Guide* and the *IBM Workplace Forms Server — API for Java User's Manual*.

Configuring a Servlet

Once you have implemented your servlet, you must create a web.xml file to provide configuration information. This file can include configuration information for your own portions of the servlet, but must also include the following *init-param* information for the IBMWorkplaceFormsServerServlet:

Setting	Mandatory	Description
translatorLocation	Yes	The location of the Translator servlet (called translator). For example: `http://www.ibm.com:8085/translator` Note that if you are setting up a clustered environment, you may need to set this to point to your clustering mechanism.
logLevel	No	Sets which events are logged, as listed: **0** — Disable logging, except for errors and exceptions. **1** — Log minimum activity, including request URIs, response events, errors, and exceptions. Default: **0**.
logServerName	No	The host name for the Log Server. Default: **localhost**.
logServerPort	No	The port number for the Log Server. Default: **4560**.

Note: If you are making changes to a web.xml file that is already deployed under WebSphere Application Server, be sure you use the Application Assembly Tool to make those changes. WAS does not automatically recognize changes made to the web.xml file through a text editor.

Deploying a Servlet

Once you have implemented and configured a servlet, you must deploy the servlet to your application server. This process requires you to:

1. Package your servlet as a WAR file.
2. Deploy the servlet to your application server.
3. If you are relying on the API for additional forms-based functionality, copy the appropriate API files to your application server.

Packaging the Servlet

When you package your servlet, follow the standard procedure for creating a WAR file. In addition, include the following files in the WEB-INF/lib/ folder:

- ws_framework.jar
- ws_common.jar
- log4j.jar

These files are installed with Webform Server in the following location:

```
<Webform Server Install Dir>\redist\
```

Note: Do not include jar files from the API in your package, even if your servlet is relying on functionality provided by the API. Instead, distribute those files as explained in ″Distributing the API″.

Deploying the Servlet

Once you have packaged your servlet, you are ready to deploy it to your application server. To do this, follow the instructions provided with the application server or servlet runner you are using.

Distributing the API

If your servlet relies on the API, you must also setup the API on your application server. For more information, refer to “Configuring WAS to Access the API” on page 59.

Creating a Portlet

If you want to implement your application within a portal, you must create one or more portlets that communicate with Webform Server. These portlets represent your overall application, and route forms between the user and the Translator when necessary.

The portlets can include any functionality you want. For example, you might create a portlet that not only uses the Translator to provide HTML forms to the user, but that also routes completed forms to the next step in their life cycle.

To create this portlet, you must extend the IBMWorkplaceFormsServerPortlet class that is provided with Webform Server. This class is based on the JSR 168 portlet standard, and also provides a number of methods that will simplify the process of implementing your portlet.

Note: If you are using XFDL forms in a portlet, your users must have Workplace Forms Viewer 2.5 or later, or a PureEdge-branded Viewer of at least version 6.2. Portlets will not work with earlier versions of the Viewer.

Normal Portlet Operations

Normal operation of a portlet with Webform Server assumes that there are two form-related actions the user might take:

1. The user requests a form from the portlet. This is generally done by clicking an action link in the portlet.
2. The user submits a form that is being displayed by the portlet. This may be either an XFDL form or a form that was previously translated into HTML.

In either case, the request is processed through a combination of functionality that is provided by the IBMWorkplaceFormsServerPortlet class and code that you write while extending the portlet. Furthermore, the IBMWorkplaceFormsServerPortlet class is designed to ignore requests that do not involve XFDL forms. This means that all other operations will fall through the IBMWorkplaceFormsServerPortlet to your custom code.

In general, you must implement a *processActionEx* and *doViewEx* method within your portlet to handle the expected form-related scenarios.

The processActionEx Algorithms

In general, the *processActionEx* algorithm must handle two scenarios. In the first scenario, the end-user clicks an action link to request a form from the portlet. In the second scenario, the end-user submits a form to the portlet. Each scenario requires you to add special processing to your portlet.

User Requests a Form

The following algorithm presents a simple way to respond to a request for a form:

1. The user clicks a link that requests an XFDL form from the portlet.
2. The request falls through the IBMWorkplaceFormsServerPortlet to your *processActionEx* method.

3. In your *processActionEx* method, you retrieve the details of the requested action from the request object. For example, you might retrieve the name of the requested file.
4. You then write those details to a session bean. Again, this may simply be the name of the requested file.
   - Once you have written the information to the session bean, you can end your *processActionEx* method. The IBMWorkplaceFormsServerPortlet object will continue operations from that point.

Further processing is carried out as part of the *doViewEx* method, which is triggered by the Portal after the *processActionEx* method is finished.

User Submits a Form

The following algorithm presents a simple way to respond to a form submission:

1. The user submits a form to the portlet.
2. If the form was previously translated into HTML, the IBMWorkplaceFormsServerPortlet automatically translates the form back to XFDL, then passes it through to your *processActionEx* method. If the form is XFDL, it is automatically passed through to your *processActionEx* method.
3. In your *processActionEx* method, you process the form as required. This may include saving the form to disk, archiving the form to a database, or sending the form to the next stage of a workflow.
4. Write the details of the response to a session bean. For example, you might write an HTML page to the session bean, or simply the location of an HTML page that should be sent to the user.
   - Once you have written the information to the session bean, you can end your *processActionEx* method. The IBMWorkplaceFormsServerPortlet object will continue operations from that point.

Further processing is carried out as part of the *doViewEx* method, which is triggered by the Portal after the *processActionEx* method is finished.

The doViewEx Algorithms

In general, the *doViewEx* method must handle two scenarios. In the first scenario, the *doViewEx* method must display a form to the user. In the second scenario, the *doViewEx* method must display some other result, such as an HTML response. Each case requires you to add special processing to your portlet.

Displaying an XFDL Form

The following algorithm presents a simple way of displaying an XFDL form:

1. The user has requested an XFDL form and the *processActionEx* has written the name of the form to a session bean.
2. The *doViewEx* method retrieves the name of the form from the session bean, and then locates and loads the form.
3. The *doViewEx* method writes the form to the response object, and sets the content type of the response object to **application/vnd.xfdl**.
   - Once you have written the form to the response object, you can end your *doViewEx* method. The IBMWorkplaceFormsServerPortlet will continue operations from that point.
4. The IBMWorkplaceFormsServerPortlet automatically detects whether the user has the Viewer, and returns the form in either XFDL or HTML as appropriate.

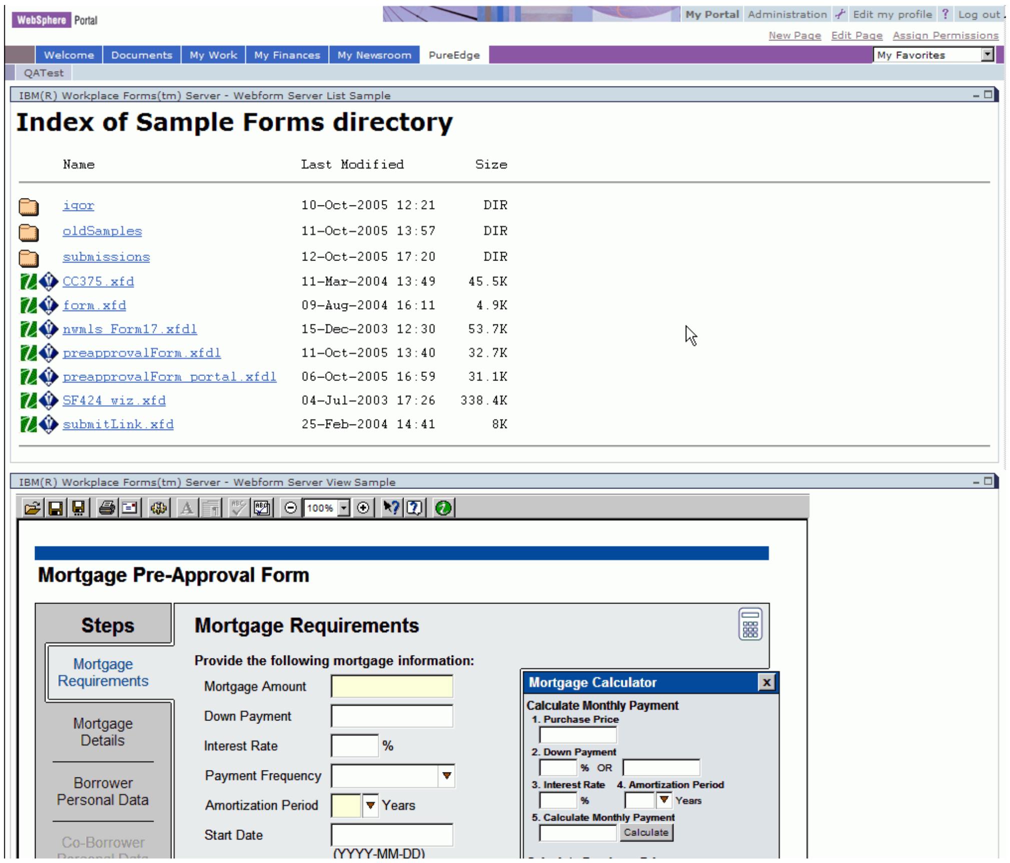
Displaying a Non-XFDL Response

The following algorithm presents a simple way of displaying a non-XFDL response:

1. The user has submitted an XFDL form, and the *processActionEx* method has written the filename of the appropriate response to a session bean.
2. The *doViewEx* method retrieves the filename of the response from the session bean, and then locates and loads the response.
3. The *doViewEx* method writes the form to the response object, and sets the content type as appropriate. For example, for an HTML response, you would set the content type to **text/HTML**.
   - Once you have written the response to the response object, you can end your *doViewEx* method. The IBMWorkplaceFormsServerPortlet will continue operations from that point.
4. The IBMWorkplaceFormsServerPortlet returns the response to the user.

Working with Multiple Portlets

In many cases, you may want to design a portal that uses several portlets. The most common example of this is a two pane portal, in which each pane is a separate portlet. The first portlet displays a list of forms, while the second portlet displays any form the user selects from the first portlet, as shown:



IBM(R) Workplace Forms(tm) Server - Webform Server List Sample

Index of Sample Forms directory

```
     Name                         Last Modified        Size

     igor                         10-Oct-2005 12:21     DIR
     oldSamples                   11-Oct-2005 13:57     DIR
     submissions                  12-Oct-2005 17:20     DIR
     CC375.xfd                    11-Mar-2004 13:49   45.5K
     form.xfd                     09-Aug-2004 16:11    4.9K
     nvmls Form17.xfdl            15-Dec-2003 12:30   53.7K
     preapprovalForm.xfdl         11-Oct-2005 13:40   32.7K
     preapprovalForm portal.xfdl  06-Oct-2005 16:59   31.1K
     SF424 viz.xfd                04-Jul-2003 17:26  338.4K
     submitLink.xfd               25-Feb-2004 14:41      8K
```

IBM(R) Workplace Forms(tm) Server - Webform Server View Sample

Mortgage Pre-Approval Form

Steps

Mortgage Requirements

Mortgage Details

Borrower Personal Data

Co-Borrower Personal Data

Mortgage Requirements

Provide the following mortgage information:

Mortgage Amount

Down Payment

Interest Rate %

Payment Frequency

Amortization Period Years

Start Date (YYYY-MM-DD)

Mortgage Calculator

Calculate Monthly Payment

1. Purchase Price
2. Down Payment % OR
3. Interest Rate % 4. Amortization Period Years
5. Calculate Monthly Payment

To create a portal with multiple portlets, you must create each portlet separately and split the overall functionality between them. You must also implement a communication protocol. This allows each portlet to send information to the others, so that they can share state information or pass user requests.

Be aware that only one portlet per page can display converted HTML forms. Displaying multiple HTML forms at the same time may cause the forms to conflict with each other and produce unpredictable results. However, this conflict does not occur if you are displaying XFDL forms, because the Viewer can run in multiple portlets on a single page without causing conflicts.

Although any number of portal designs are possible, we will focus on the two pane portal that we have already mentioned. This will introduce you to the concepts you need to understand to create more sophisticated portals.

To create the two pane portal, you must do the following:
- Devise a communication method for the portlets.
- Create a list portlet, which displays a list of the available forms.
- Create a form portlet, which displays a particular form when the user selects one from the list.

Devising a Communication Method

When working with multiple portlets, you will often need them to communicate with each other. For example, in our two pane portlet, the list portlet must be able to tell the form portlet which form the user has selected.

There are a number of ways to accomplish this, but all of them rely on using the portal session to store global information. For instance, you might create an external manager class that could track information from various portlets, or you might code the portlets to directly access global information each time they are run.

If we use the direct method for our two pane portlet, we can have the list portlet write the name of the selected form to the session as part of its *processActionEx* method. The form portlet can then read that information from the session as part of its *doViewEx* method, and load the appropriate form for display.

The rest of this discussion will assume that this is the communication method being used.

Creating a List Portlet

The list portlet displays a list of the forms available to the user. It may display a predetermined list or a list that is generated at run-time. In any case, the user can select a form from the list which will then be displayed by the form portlet.

Because this portlet will not handle XFDL forms directly, it does not need to extend the IBMWorkplaceFormsServerPortlet class. Instead, it will extend the GenericPortlet class, and as such requires a *doView* method and a *processAction* method.

The doView Method

The *doView* method must generate the list of the forms that are available to the user. The following algorithm presents a simple *doView* method for this portlet:

1. The *doView* method creates a list of forms available.
   - This list may be created in a number of ways. For example, the list may be hard-coded into the portlet, may be generated at run time by searching some directories, or may be generated through another method.
2. The *doView* method then writes the list to the response object, typically as part of an HTML page, and sets the appropriate content type, such as **text/HTML**.

   - Once you have written the response to the response object, you can end your *doView* method. The GenericPortlet will continue operations from that point.
3. The GenericPortlet returns the response to the user.

The processAction Method

The *processAction* method must write the name of the selected form to the portal session, so that the form portlet can retrieve the name and load the correct form. The following algorithm presents a simple *processAction* method for this portlet:

1. The user clicks a link to one of the forms listed in the portlet.
2. The request falls through the GenericPortlet to your *processActionEx* method.
3. The *processAction* method determines which form was requested by checking the request object.
4. The *processAction* method writes the location of the requested form to the portal session.
   - This information will be retrieved from the session by the form portlet.
   - Once you have written the information to the session bean, you can end your *processAction* method. The GenericPortlet object will continue operations from that point.

Creating a Form Portlet

The form portlet displays any form that the user selects from the list portlet, and updates whenever the user selects a new form from the list portlet.

Since this portlet will handle XFDL forms, it must extend the IBMWorkplaceFormsServerPortlet class. As such, this portlet requires a *doViewEx* method and a *processActionEx* method. This portlet also requires a *doView* method, for reasons that will be explained later.

The doViewEx Method

The *doViewEx* method must retrieve the name of the requested form from the portal session, and display that form to the user. The following algorithm present a simple *doViewEx* method for this portlet:

1. The user has selected an XFDL form in the list portlet, and that portlet has written the name of the form to the portal session.
2. The *doViewEx* method retrieves the name of the form from the portal session, and then locates and loads the form.
3. The *doViewEx* method writes the form to the response object, and sets the content type of the response object to **application/vnd.xfdl**.
   - Once you have written the form to the response object, you can end your *doViewEx* method. The IBMWorkplaceFormsServerPortlet will continue operations from that point.
4. The IBMWorkplaceFormsServerPortlet automatically detects whether the user has the Viewer, and returns the form in either XFDL or HTML as appropriate.

The processActionEx Method

The *processActionEx* method must accept forms that the user submits and process them properly. The following algorithm presents a simple *processActionEx* method for this portlet:

1. The user submits a form to the portlet.
2. If the form was previously translated into HTML, the IBMWorkplaceFormsServerPortlet automatically translates the form back to

XFDL, then passes it through to your *processActionEx* method. If the form is XFDL, it is automatically passed through to your *processActionEx* method.

3. In your *processActionEx* method, you process the form as required. This may include saving the form to disk, archiving the form to a database, or sending the form to the next stage of a workflow.
4. Write the details of the response to a session bean. For example, you might write an HTML page to the session bean, or simply the location of an HTML page that should be sent to the user.
   - Once you have written the information to the session bean, you can end your *processActionEx* method. The IBMWorkplaceFormsServerPortlet object will continue operations from that point.

The doView Method

The IBMWorkplaceFormsServerPortlet maintains a state of either empty or full. The empty state means that no form is loaded, and that *doViewEx* should be called to load a form. The full state means that a form is loaded and that *doViewEx* does not need to be called.

However, in our two pane scenario, this means that a form is loaded only when the user first selects a form. If the user selects another form after that, the form portlet is not updated because *doViewEx* is never called.

To correct this, you must overwrite the *doView* function and force the portlet to refresh when a new form is selected. This will put the portlet in its empty state, which will cause the portlet to call the *doViewEx* method.

The following algorithm demonstrates how to overwrite the *doView* function:

1. Check the session to see if the user has selected a new form in the list portlet.
2. If the user has selected a new form, refresh the form portlet by calling *resetPortlet*.
   - The *resetPortlet* method is part of the IBMWorkplaceFormsServerPortlet class.
3. Call the super method. For example:
   ```
   super.doView()
   ```
   - Once you have called the super, you can end your *doView* method. The IBMWorkplaceFormsServerPortlet object will continue operations from that point, and will call *doViewEx* immediately after *doView* has finished processing.

Implementing a Portlet

You must implement your portlet by extending the IBMWorkplaceFormsServerPortlet class, which is provided as part of Webform Server. The IBMWorkplaceFormsServerPortlet class is itself an extension of the standard HTTP portlet class defined in javax.portlet.GenericPortlet. As such, it inherits the methods defined for a generic portlet, and you are free to use or overwrite those methods.

In addition, the IBMWorkplaceFormsServerPortlet class provides a collection of custom methods to assist you in writing your portlet. These methods provide form-related functionality, such as setting the response type to XFDL or HTML, determining whether the client has the Viewer installed, and so on.

For more information about the IBMWorkplaceFormsServerPortlet class and its methods, refer to the Webform Server Javadocs on the IBM website.

Some sample portlets are also provided with Webform Server, including the source code used to implement the portlets. For more information, refer to “Setting Up the Portlet Sample” on page 51.

Setting Up Your System to Use the IBMWorkplaceFormsServerPortlet Class

Before you can use the IBMWorkplaceFormsServerPortlet class, you must add the ws_framework.jar, ws_common.jar, and log4j.jar to your classpath. These files are installed with Webform Server in the following location:

```
<Webform Server Install Dir>\redist\
```

Using Other Form API Methods During Development

You may want your portlet to access XFDL forms directly. For example, you may want to prepopulate forms as you load them, or extract data from forms when they are submitted. To do this, you must install the API and use it in conjunction with the IBMWorkplaceFormsServerPortlet class to develop your portlet.

For more information about the API, refer to the *IBM Workplace Forms Server — API Installation & Setup Guide* and the *IBM Workplace Forms Server — API for Java User’s Manual*.

Configuring a Portlet

Once you have implemented your portlet, you must create a portlet.xml file to provide configuration information. This file can include configuration information for your own portions of the portlet, but must also include the following *init-param* information for each unique IBMWorkplaceFormsServerPortlet you are using:

Setting	**Mandatory**	**Description**
translatorLocation	Yes	The location of the Translator servlet (called translator). For example: `http://www.PureEdge.com:8085/translator`
logLevel	No	Sets which events are logged, as listed: **0** — Disable logging, except for errors and exceptions. **1** — Log minimum activity, including request URIs, response events, errors, and exceptions. Default: **0**.
logServerName	No	The host name for the Log Server. Default: **localhost**.
logServerPort	No	The port number for the Log Server. Default: **4560**.
portletWidth	No	The width of the portlet in the browser. Your portlet should be no larger than the screen. This ensures that the Webform Server toolbar can never scroll off the screen and be inaccessible to the user. Default: **640**.

Setting	Mandatory	Description
portletHeight	No	The height of the portlet in the browser. Your portlet should be no larger than the screen. This ensures that the Webform Server toolbar can never scroll off the screen and be inaccessible to the user. Default: **480**.
viewerTTL	No	The ″time to live″ for the Viewer, in seconds. This is used to determine how long the Viewer is maintained in the background if the user navigates to a different page. Default: **15**.

Note: WebSphere Application Server will not recognize changes you make to a portlet.xml file that is already deployed. If you need to make changes to your portlet, you must change the portlet.xml file, repackage it in you WAR file, and redistribute your war file to the portal server.

Configuring a Portlet for WebSphere Portal

If you are going to use your portlet with WebSphere Portal, your portlet.xml file must also contain the following settings:

supported-locale
This defines the support languages for the portlet. For example:

```
en
```

resource-bundle
This defines the location of the language resource bundle. Set it to:

```
i18n.<name>
```

Choose a name that uniquely identifies the resource bundle. This name will also be used as part of the filenames for the resource properties files.

Using the name defined for the resource-bundle setting above, you must also create the following two properties files:

- *<name>*_*<locale>*.properties
- *<name>*Text_*<locale>*.properties

The <*locale*>.properties File

The *<name>*_*<locale>*.properties file is a simple text file that contains tag-value pairs. It must contain the following settings:

javax.portlet.title
The fullname to assign to the portlet. This title appears on the portlet. For example:

```
IBM Form View Sample
```

javax.portlet.short-title
The short name to assign to the portlet. For example:

```
IBMFormView
```

javax.portlet.keywords
One or more keywords that can be used to search for the portlet. For example:

```
IBM
```

The Text_<*locale*>.properties File

The <*name*>Text_<*locale*>.properties file is a simple text file that you can leave empty, but that you must include for the portlet to run properly under WebSphere Portal.

Configuring the WebformPortletForwarder Servlet

When using portlets with Webform Server, you must also use a special servlet, called the WebformPortletForwarder. This servlet is provided as part of the Webform Server installation, and handles specific requests, such as printing or saving the form, that the portlets cannot handle themselves.

To ensure that your application uses the WebformPortletForwarder Servlet properly, you must add the servlet to your portlet's web.xml file.

Adding the Servlet

To add the servlet, open the web.xml file and add the following information. Note that the init-params are defined in the next section.

```
<servlet>
   <servlet-name>WebformPortletForwarder</servlet-name>
   <servlet-class>com.ibm.form.webform.pws.portlet.jsr.WebformPortletForwarder
   </servlet-class>
   <init-param>
      Init Params
   </init-param>
</servlet>
```

Configuring the Servlet

Once you have added the WebformPortletForwarder servlet to the web.xml file, you can configure a number of options. These are configured as init-params in the web.xml file, and are defined below:

Setting	Mandatory	Description
translatorLocation	Yes	The location of the Translator servlet (called translator). For example: `http://www.ibm.com:8085/translator` Note that if you are setting up a clustered environment, you may need to set this to point to your clustering mechanism.
logLevel	No	Sets which events are logged, as listed: **0** — Disable logging, except for errors and exceptions. **1** — Log minimum activity, including request URIs, response events, errors, and exceptions. **2** — Log all activity, including complete requests, complete responses, errors, and exceptions. Default: **0**.
logServerName	No	The host name for the Log Server. Default: **localhost**.
logServerPort	No	The port number for the Log Server. Default: **4560**.

Mapping the Servlet

Finally, you must map the WebformPortletForwarder servlet. To do this, add the following information to the web.xml file:

```
<servlet-mapping>
   <servlet-name>WebformPortletForwarder</servlet-name>
   <url-pattern>/PWSForwarder/*</url-pattern>
</servlet-mapping>
```

Example of a Complete web.xml File

The following code shows what a complete web.xml file might look like once the WebformPortletForwarder has been added:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE web-app web-app PUBLIC
   "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebformPortletSamples">
   <display-name>WebformSamplePortlets</display-name>
   <servlet>
      <servlet-name>WebformPortletForwarder</servlet-name>
      <servlet-class>com.ibm.form.webform.portlet.jsr.WebformPortletForwarder
  </servlet-class>
      <init-param>
         <param-name>translatorLocation</param-name>
         <param-value>http://localhost:8085/translator</param-value>
      </init-param>
   </servlet>
   <servlet-mapping>
      <servlet-name>WebformPortletForwarder</servlet-name>
      <url-pattern>/PWSForwarder/*</url-pattern>
   </servlet-mapping>
</web-app>
```

Deploying a Portlet

Once you have implemented and configured a portlet, you must deploy the portlet to your application server. This process requires you to:

1. Package your portlet as a WAR file.
2. Deploy the portlet to your portal server.
3. If you are relying on the API for additional forms-based functionality, copy the appropriate API files to your application server.

Packaging the Portlet

You package your portlet by creating a standard WAR file. In addition, you must include a number of Webform Server files.

To add the Webform Server files to your WAR file:

1. Locate the following directory:

   `<Webform Server Install Dir>\redist\`
2. Copy the following files from that directory to the WEB-INF/lib/ directory in your WAR file:
   - ws_framework.jar (the framework includes the WebformPortletForwarder servlet)

- ws_common.jar
- log4j.jar

3. Copy the following configuration files are in the WEB-INF directory of your WAR file:
   - web.xml (as written by you for the WebformPortletForwarder servlet)
   - portlet.xml (as written by you for your portlets)

   Note: Do not include jar files from the API in your package, even if your servlet is relying on functionality provided by the API. Instead, distribute those files as explained in ″Distributing the API″.

Deploying the Portlet

Once you have packaged your portlet, you are ready to deploy it to your portal server. To do this, follow the instructions provided with the application server or servlet runner you are using.

Distributing the API

If your portlet relies on the API, you must also setup the API on your application server. For more information, refer to “Configuring WAS to Access the API” on page 59.

Working with the Samples

Your Webform Server installation includes a servlet sample and a portlet sample that you can deploy to your application server or portal server. These samples are designed to demonstrate a simple forms-based application, and includes source code that you can review.

About the Samples

Both samples present a simple interface that allows you to view the contents of a particular directory on your computer. The sample will list the forms in that directory, and will allow you to view any form you select, and submit any form you view.

The servlet accomplishes this by first presenting a view of the directory. When you select a form, the form replaces the directory in the browser. When the user submits the form, the directory is again displayed.

The portlet sample accomplishes this by using two portlets. The first portlet displays a list of the forms in the directory, while the second portlet displays the selected form.

In both samples, users can navigate to a sub-directory that stores submitted forms, and can review the sample code and forms as they like. Particular areas that may be of interest include how globalization was accomplished in the portlet and servlet and how XForms submissions of type ″replace=’all’″ is handled in the XForms sample form.

Setting Up the Sample Servlet

There are two different procedures for setting up the sample servlet, depending on whether you want to run the servlet on the same server as the Translator, or whether you want to run it on a different application server.

Running the Sample Servlet on the Translator Server

The sample servlet is installed with Webform Server, and will automatically run on the Translator server if you accepted all of the default settings during the installation process.

If you did not accept the default settings, you should review the configuration settings, as explained in ″Configuring a Servlet″.

Deploying the Sample Servlet on a Different Application Server

The sample servlet is set up to be run on the same server as the Translator. If you want to run it on a different server, you must:

- Modify the sample servlet.
- Deploy the updated WAR file to your application server.
- Setup the API to work with your application server.

Modifying the Sample Servlet

Before you can modify the sample servlet, you must ensure that you have a version of the Java Development Kit installed on your computer. This is used to update the WAR file once you have made changes to the servlet.

To modify the sample servlet:

1. Locate the following file:
   ```
   <Webform Server Install Dir>/Samples/Servlet/SampleSrc.zip
   ```
2. Unzip the file.
3. In the unzipped directory, locate the following file:
   ```
   WEB-INF/web.xml
   ```
4. Open the file in a text editor and update the following init-params:
   - translatorLocation
   - logServerName
   - logServerPort
   - For more information about these parameters, see ″Configuring a Servlet″.
5. Save your changes.
6. Run the createWAR.bat file in the root of the unzipped directory.
   - This creates a new WAR file called Sample.war that contains the sample servlet.

Deploying the Sample Servlet

To deploy the sample servlet you must install the WAR file you just created on your application server. For more information about installing a WAR file, refer to the documentation for the application server you are using.

Setting Up the API

This process varies depending on the application server you are using. For detailed instructions in setting up the API with WebSphere Application Server, refer to "Configuring WAS to Access the API" on page 59.

Using the Sample Servlet

The sample servlet provides a simple interface for working with forms on a server. Using this application, you can:

- Open a blank form in XFDL or HTML.
- Submit a completed form.
- Open a previously submitted form in XFDL or HTML.

The following sections explain the functionality of the servlet.

Starting the Servlet

To start the sample servlet, open a browser and go to the following URL:

```
http://<server name>:<port>/Samples
```

Note: You must supply the name of the server that is running the sample servlet, as well as the appropriate port number.

This loads a welcome page that gives you the option of running the servlet or reviewing the source code used to create it. Once you run the sample, the servlet loads a directory listing of sample forms and displays a list of all of the forms available.

Opening a Template Form

Once the sample is running, you will see a directory listing that includes all template forms available to the application. The directory listing provides three ways to open a form:

- **Force XFDL —** Clicking the Force XFDL icon will open the form as an XFDL document. You must have the Viewer installed to view the document in this manner.
- **Force HTML —** Clicking the Force HTML icon will open the form as an HTML document, which will be displayed in your web browser.
- **Click the link —** Clicking the link to the form will cause the sample application to automatically detect whether you have the Viewer installed. If you do, the form will open as XFDL; otherwise, the form will open as HTML.

While the sample only includes one template form, you can add further forms to the sample by copying forms to the following directory:

```
<Servlet Deployment Dir>\ServletSampleForms
```

Submitting a Form

When you submit a form to the sample, the form portlet writes your forms to a submissions folder within the forms directory.

Opening a Submitted Form

Once you have submitted one or more forms to the sample, the servlet will include a submissions sub-directory in its listing of the available forms.

To open a form you have already submitted, simply navigate to this directory (by clicking it) and select the form you want to open. As with the template forms, you can open previously submitted forms as either XFDL or HTML.

Deleting a Submitted Form

To delete a form you have already submitted, you must manually delete the form from the form submissions directory. For example, under Tomcat this directory is:

```
<Servlet Deployment Dir>\ServletSampleForms\submissions
```

Setting Up the Portlet Sample

There are two different procedures for setting up the portlet sample, depending on whether you want to run the portlets on the same server as the Translator, or whether you want to run them on a separate Portal Server.

Running the Portlet Sample on the Translator Server's Server

The portlet sample is included with Webform Server as a WAR file that contains two portlets. This sample assumes that you are running the Translator Server on the same server as the portlet sample, so does not need to be modified. However, before you use the sample, you must install it in your Portal Server:

1. Locate the WebformSamplePortlet.WAR file in the following directory:

```
<Webform Server Installation Dir>/Samples/Portlet/
```

2. Install the WAR file into your Portal Server.
   - Consult the documentation for your portal software to determine how to do this.
3. Create a page that uses the following portlets:
   - **IBM Workplace Forms Server - Webform Server List Sample —** This portlet displays a list of the available forms.
   - **IBM Workplace Forms Server - Webform Server View Sample —** This portlet displays the form selected by the user.

Once you have created a page with the portlets, you can access that page and use the sample. For more information about using the sample, refer to "Using the Portlet Sample" on page 53.

Deploying the Portlet Sample on a Separate Portal Server

The portlet sample is set up to be run on the same server as the Translator. If you want to run it on a different server, you must:

- Modify the portlet sample.
- Deploy the updated WAR file to your portal server.
- Setup the API to work with your portal server.

Modifying the Portlet Sample

To modify the portlet sample:

1. Locate the following file:
   ```
   <Webform Server Install Dir>/Samples/Portlet/WebformPortletSampleSrc.zip
   ```
2. Unzip the file.
3. In the unzipped directory, locate the following file:
   ```
   WEB-INF/portlet.xml
   ```
4. Open the file in a text editor and update the following *init-params*:
   - translatorLocation
   - logServerName
   - logServerPort
   - For more information about these parameters, see ″Configuring a Portlet″.
5. Save your changes.
6. In the unzipped directory, locate the following file:
   ```
   WEB-INF/web.xml
   ```
7. Open the file in a text editor and update the following *init-params*:
   - TranslatorLocation
   - LogServerName
   - For more information about these parameters, see ″Configuring a Servlet″.
8. Save your changes.
9. Run the createWAR.bat file in the root of the unzipped directory.
   - This creates a new WAR file called WebformSamplePortlet.war that contains the portlet sample.

Deploying the Portlet Sample

1. Ensure that the WebformSamplePortlet.WAR file is available to your portal server.

   - You may need to copy the file to the server that is running your portal server.
2. Install the WAR file into your portal server.
   - Consult the documentation for your portal software to determine how to do this.
3. Create a page that uses the following portlets:
   - **Webform Server Forms List Sample —** This portlet displays a list of the available forms.
   - **Webform Server Form Viewer Sample** — This portlet displays the form selected by the user.

Setting Up the API

This process varies depending on the application server you are using. For detailed instructions in setting up the API with WebSphere Application Server, refer to “Configuring WAS to Access the API” on page 59.

Using the Portlet Sample

The portlet provides a simple interface for working with forms on a server. Using this application, you can:

- Open a blank form in XFDL or HTML.
- Submit a completed form.
- Open a previously submitted form in XFDL or HTML.

The following sections explain the functionality of the portlet.

Starting the Portlet Sample

To start the portlet sample, open the appropriate page in your Portal. One portlet will display a list of the available forms, while the other portlet will contain nothing.

Opening a Template Form

Once the sample is running, you will see a directory listing that includes all template forms available to the application. The directory listing provides three ways to open a form:

- **Force XFDL —** Clicking the Force XFDL icon will open the form as an XFDL document. You must have the Viewer installed to view the document in this manner.
- **Force HTML —** Clicking the Force HTML icon will open the form as an HTML document, which will be displayed in your web browser.
- **Click the link —** Clicking the link to the form will cause the sample application to automatically detect whether you have the Viewer installed. If you do, the form will open as XFDL; otherwise, the form will open as HTML.

While the sample only includes one template form, you can add further forms to the sample by copying forms to the following directory:

```
<Portlet Deployment Dir>\SampleForms
```

Submitting a Form

When you submit a form to the sample, the form portlet writes your forms to a submissions folder within the forms directory.

Opening a Submitted Form

Once you have submitted one or more forms to the sample, the list portlet will include a submissions sub-directory in its listing of the available forms.

To open a form you have already submitted, simply navigate to this directory (by clicking it) and select the form you want to open. As with the template forms, you can open previously submitted forms as either XFDL or HTML.

Deleting a Submitted Form

To delete a form you have already submitted:

1. In the list portlet, navigate to the submissions directory.
2. To the right of the file you want to delete, click **X**.

   Note: The forms stored in this directory may be automatically deleted when you restart the portlet.

Maintaining Webform Server

In general, Webform Server requires little maintenance because it supports automatic memory and file management. However, there are several tasks you should carry out to ensure that Webform Server runs optimally.

Checking the Log Files

Periodically check your log files for errors. This may help alert you to problems that were either unnoticed or unreported by your user base. For more information about the log files, "Working with the Log Server" on page 27.

Periodically Check the Webform Server Services

Periodically check your server to ensure that the Webform Server services are still running properly. The steps required to do this differ depending on your operating system.

Windows

To check the services on Windows:

1. Open the Windows Services monitor. For example, on Windows 2003 Server you would open:

   Start → **Settings** → **Control Panel** → **Administrative Tools** → **Services**
2. Look for the following entries in the Services list:
   - **Workplace Forms Webform Translator —** This means the Translator is running on the server.
   - **Workplace Forms Webform Access Controller** — This means Access Control Server is running on the server.
   - **Workplace Forms Webform Logger —** This means the Log Server is running on the server.

If you do not see all of the services you expect, reboot the server and check the Windows Services monitor again. The service should start automatically on reboot.

Linux

To check the services on Linux:

1. List the running java processes, as shown:
   ```
   ps -A uwww|grep java
   ```
2. Look for the following entries in the process list:
   - **Access Control Server**: com.ibm.form.webform.accesscontrol.server.AccessControlServer
   - **Log Server**: com.ibm.form.webform.logging.log4j.server.LogServer
   - **Translator**: org.apache.catalina.startup.Bootstrap

If you do not see all of the processes you expect, reboot the server and check the running java processes again. The processes should start automatically on reboot.

Extending Webform Server

You can create extensions for Webform Server that allow it to do custom processing on the server-side. For security reasons, Webform Server extensions cannot be embedded in a form, published from a user's computer, or effect the user interface. Both Java and C extensions are supported.

Publishing Extensions

There are two ways in which you can publish your extensions (ifx files) to your Webform Server application:

- The Webform Server extensions folder:
  - **Windows**: C:\Program Files\IBM\Workplace Forms\Server\xx\Webform Server\Translator\TranslatorApp.ear\ translator.war\extensions
  - **UNIX**: /opt/IBM/Workplace_Forms/Server/xx/ Webform_Server/ Translator/TranslatorApp.ear/translator.war/extensions.
- Any directory specified by the **extensionsDirectory** property defined in the translator.properties file.

Note: As your extensions are run on the server-side, they should be thoroughly tested to ensure that they perform as expected and do not unintentionally overload the server. For detailed information on creating extensions, see the *Workplace Forms Server - API User's Manual* for Java or C.

Configuring WAS to Access the API

The Translator component of the Webform Server should have a dedicated WAS to ensure its best performance. It should not share WAS with other applications, such as Portal Server. However, you may want other web applications, such as portlets and servlets, to be able to access Workplace Forms Server — API. The best way to allow other applications to access the API is to install it on the same computer. These applications should not access the API installed with Webform Server. You can then configure WAS to use the API.

1. Install and configure Workplace Forms Server — API.
   - For detailed instructions, see the *IBM Workplace Forms Server — API Installation and Setup Guide*.
2. Set the WebSphere variables (see "Setting the WebSphere Variables").
3. Modify the PureEdgeAPI.ini file (see "Creating the PureEdgeAPI.ini File").
4. Set the Java process definitions (see "Setting the Java Process Definitions" on page 60).
5. Set the shared libraries (see "Setting the Shared Libraries" on page 60).
6. Set the server class loader (see "Setting the Server Class Loader" on page 61).

Setting the WebSphere Variables

1. In the **WebSphere Application Server Administration Console**, go to **Environment → WebSphere Variables|Manage WebSphere Variables**.
2. To set the scope, click **Browse Nodes** and select the node that contains the server you are using.
   - For example, **WebSphere_Portal** or **server1**.
3. Click **Apply**.
4. Click **New** to create an environment variable.
5. Under **Name**, type **WFS_API_DIR**.
6. Under **Value**, type the directory in which you installed the API files.
   - For example, `C:\Program Files\IBM\Workplace Forms\Server\2.6\API\redist\msc32\`
7. Repeat steps 4-5 to create another environment variable called **WFS_API_LIB_DIR** and point it to the API's Java libraries directory.
   - For example, ${WFS_API_DIR}/PureEdge/70/java/classes

Creating the PureEdgeAPI.ini File

Note: You may have already completed this task as part of the API install.

1. In a text editor, create a file called PureEdgeAPI.ini.
2. Include the following two lines:
   ```
   [API]
   *=<API Install Dir>\redist\msc32\PureEdge\xx
   ```
   Note that you should replace <*API Installation Dir*> with the actual path to the API installation directory and *xx* with the version number of the API.
   - For example: `*= C:\Program Files\IBM\Workplace Forms\Server\2.6\API\redist\msc32\PureEdge\70`

For more information about this file, refer to the description of the *initialize* method in the *IBM Workplace Forms Server — API: Java User's Manual*.

3. Save the PureEdgeAPI.ini file to the <*WIN*> (Windows) or *etc* (UNIX) directory that contains the API files.

Setting the Java Process Definitions

1. Open one of the following:
   a. In WAS 6.0.2, open **Servers** → **Application servers** → **<*server_name*>** → **Java and Process Management** → **Process Definition** → **Environment Entries | Custom Properties**.
   b. In WAS 5.1.1, open **Servers** → **Application servers** → **<*server_name*>** → **Process Definition** → **Environment Entries**.
2. In this location, add or update the following properties:
   - On Windows:
     – Add a **PATH** property and point it to the API directories that contain .dll files.
       - For example, ${WFS_API_DIR};${WFS_API_DIR}/PureEdge/70/system
   - On AIX:
     – Add a **LIBPATH** property and point it to the API directories that contain .so files.
       - For example, ${WFS_API_DIR}:${WFS_API_DIR}/PureEdge/70/system
   - On Linux:
     – Add a **LD_LIBRARY_PATH** property and point it to the API directories that contain .so files.
       - For example, ${WFS_API_DIR}:${WFS_API_DIR}/PureEdge/70/system
   - On Solaris:
     – Add a **LD_LIBRARY_PATH** property and point it to the API directories that contain .so files.
       - For example, ${WFS_API_DIR}:${WFS_API_DIR}/PureEdge/70/system

Note: If the **LD_LIBRARY_PATH**, **LIBPATH**, or **PATH** properties have already been created, simply add the API directory paths to the existing properties. Remember to use the correct separator; Windows uses a semicolon while UNIX platforms uses a colon.

Setting the Shared Libraries

1. In the **WebSphere Application Server Administration Console**, go to **Environment** → **Shared Libraries**.
2. To set the scope, click **Browse Nodes** and select the node that contains your server.
3. Clear the **Server** field and click **Apply**.
4. Click **New** to create a shared library.
5. Under **Name**, type **WFS_API_LIB**.
6. Under **Classpath**, list the following files (one per line):
   - ${WFS_API_LIB_DIR}/pe_api.jar
   - ${WFS_API_LIB_DIR}/pe_api_native.jar
   - ${WFS_API_LIB_DIR}/uwi_api.jar
   - ${WFS_API_LIB_DIR}/uwi_api_native.jar

7. Click **OK**.

Setting the Server Class Loader

1. Open one of the following:
   a. In WAS 6.0.2, open **Servers** → **Application servers** → **<*server_name*>** → **Java and Process Management** → **Classloader**.
   b. In WAS 5.1.1, open **Servers** → **Application servers** → **<*server_name*>** → **Classloader**.
2. If there is already a classloader for your application server, go to step 5. Otherwise, go to step 3.
3. To create a new classloader, click **New**.
4. Under **Class loading mode**, select **Parent Last** and click **OK**.
5. Select the classloader for your application server and click **Libraries**.
6. Click **Add**.
7. Under **Library Name**, select **WFS_API_LIB**.
8. Click **OK**.
9. Save all changes to the master configuration.

Troubleshooting

This section provides general troubleshooting information for Webform Server. For information specific to troubleshooting your installation, see "Troubleshooting the Installation" on page 17.

User Errors

The following is a list of error messages that users might encounter, and suggested resolutions for those errors:

- An error has occurred! Connection refused: connect

  This message is produced when the servlet is unable to connect to the Translator. This can occur if the Translator is not available (for example, if the Translator server is down). To correct this problem, ensure that the Translator is running properly, and that the servlet can connect to the Translator through your network.
- Stat of file or path failed.

  This message is produced when the file that the user is working with is no longer available in the shared file cache (in other words, when Webform Server has deleted its copy of the form). This may occur if the *cacheExpiry* setting for the Access Control Server is less than the *contactFrequency* setting for the Translator. For more information about the *cacheExpiry* setting, refer to "Access Control Server Properties" on page 21.
- This portlet is unavailable.

  If you receive this error message while accessing the View Sample, it is likely that the API is not properly installed into the WPS portal container. To correct this problem, reinstall the API in WPS.

  If you receive this error message after clicking a link in the List Sample, the Webform Server Translator can't be found. To correct this problem ensure that the **translatorLocation** setting is pointing to a valid Translator and that the Translator is running.

Additionally, users might encounter the following general problems:

- Incorrect layout in the form or incorrect sizing of fonts, particularly when printed.

  This problem can occur if you are running the Translator component on a Linux server and have not installed the correct fonts. In this case, Webform Server will use a default font instead, which may not produce the correct layout for the form. To correct this problem, install the fonts that are used by your forms. For more information, see "Linux: Installing Fonts" on page 15.
- Delays when opening forms from users' desktops.

  If users open forms from the desktop rather than the server, it will take longer than normal to open the form. This delay occurs because Webform Server must first copy the form from the user's computer to the server. This delay will be more noticeable if the user has a slow connection or is opening a large form.

Server Errors

The following is a list of error messages that might appear in the Translator's error log, and suggested resolutions for those errors:

- com.PureEdge.error.UWIException: Unable to process xsl: ClientAbortException: java.net.SocketException: Connection reset by peer: socket write error

  You can safely ignore this error. It may appear during regular operation of Webform Server, but does not affect the operation of Webform Server or the end-user experience.
- ClientAbortException: java.net.SocketException: Connection reset by peer: socket write error

  You can safely ignore this error. It may appear during regular operation of Webform Server, but does not affect the operation of Webform Server or the end-user experience.

You may also encounter the following:
- If you are running Webform Server on Linux and you use a bold Franklin Gothic Book Font, you will see the following problems:
  - The form items containing that font will be sized for a bold and italic version of that font.
  - When printing, the items containing that font will display the bold and italic version of that font.

  To correct this problem, remove the italic version of the Franklin Gothic Book font from your server. Note that this solution will prevent the display of an italic version of this font.
-

Portal Errors

Users might encounter the following issues:
- Webform Server requires that the WebSphere Portal Server use UTF-8 encoding. For the most part, this is the default setting on the Portal server. However, there are some exceptions, which include:
  - Simplified Chinese
  - Traditional Chinese
  - Japanese
  - Korean

  If you are deploying in these languages and receive a UTF-8 encoding error, update the encoding for these languages to UTF-8 in the Portal Server Administration panel. For more information, see UTF-8 Error on Portal Server.
- If your portlet is sized so that it is larger than the screen, the user will be able to scroll the Webform Server toolbar off the top of the form. In this case, the user may then leave the focus in the lower part of the form, and scroll back up to the toolbar so that the item with the focus scrolls off the bottom of the form. If the user then clicks a button in the toolbar, the action will trigger but the form will also immediately scroll back down so that the item with the focus is visible. While this does not stop the form from functioning properly, it may annoy your users. You can prevent this problem by ensuring that the portlet is never larger than the screen.

Browser Issues

Users may encounter the following issues:
- Due to default settings in Internet Explorer, printed web pages (including forms) display the URL of the web pages in the footer.

  If the URL is longer than the width of the page, it appears to be truncated.

To prevent the URL form printing, make the following adjustments to users' computers:

1. Open their Internet Explorer browser
2. From the **File** menu, select **Page Setup**.
   – The Page Setup dialog box appears.
3. Under **Footer**, delete the **&u** characters.
4. Click **OK**.

Note: If you want to prevent the printing of all header and footer information, delete all of the characters in both the Header and Footer fields.

- Webform Server will not work properly if IE is configured to disable active scripting. Note that active scripting is normally disabled if you are running IE in high security mode. Additionally, pop-up blockers will prevent the about box from opening. However, they will not affect any other form functions.
- Users who configure their computer to display large-sized fonts (instead of system standard-sized fonts), may find that tiny dots appear when a form is displayed. Users can avoid the appearance of these dots by selecting a standard system font size.
- Due to a bug in Internet Explorer 6.0, Windows 98 computers may be caching form pages when they should not. You can resolve this issue by applying Service Pack 1 and the Cumulative Patch (available through Windows update) to the affected computers.

  For more information, refer to the Microsoft® Knowledge Base article: 321722 – Content with "Content-Encoding: gzip" Is Always Cached Although You Use "Cache-Control: no-cache".
- If you change the filename while saving a form to disk, Webform Server will not preserve the changed filename the next time you save the form.
- Using anything other than the standard Windows desktop scheme may affect the sizing of some items in your form. This occurs because different schemes use different widths for scroll bars on items. To ensure proper sizing, use the Standard Windows desktop scheme, or adjust the scroll bar width to 16.

  Under Windows XP, you can adjust the scroll bar width by right-clicking the Desktop, and selecting **Properties** → **Appearance** → **Advanced** → **Scrollbars**.
- If you uninstall the Viewer and then attempt to open a form as XFDL (rather than HTML), you may encounter an error informing you that IE does not know how to open the document. This occurs because the .xfd, .xfdl, and .frm extensions are still associated with the Viewer, but the Viewer is no longer available to open the form.

  To correct this problem, you must delete the associations for those extensions, as follows:

  1. Open Windows Explorer.
  2. From the **Tools** menu, select **Folder Options**.
  3. In the Folder Options dialog box, select the **File Types** tab.
  4. Scroll through the list of extensions until you find FRM.
  5. Repeat steps 4 and 5 for the XFD and XFDL extensions.
- If you are working in a portlet and you link to a document from a form, then submit the original form, you will be returned to the portal's login page.
- If you have difficulties printing forms on a Windows 98 computer, you can resolve this issue by applying Service Pack 1 and the Cumulative Patch (available through Windows update) to the affected computers.

Accessibility

Users with disabilities can still work with Webform Server forms by taking advantage of the following features:

- Support for the JAWS screen reading utility.
- Respect for changes to Operating System colors (including High Contrast mode).
- Aids to Navigating forms.

Support for JAWS

For persons with vision disabilities, Webform Server supports version 5.0 of the JAWS screen reader from Freedom Scientific. To enable the screen reader, the user must do the following:

1. Enable the accessibility mode by clicking **Accessibility Mode** in the toolbar or typing CTRL + H.
2. Activate ″form mode″ in JAWS.

Once in form mode, JAWS will read each item on the form as it gains the focus, as well as any special accessibility messages in the form.

For more information on JAWS announcements and proper accessibility practices when designing forms, refer to the document *″Best Practices for Designing Forms for Webform Server″* available from IBM.

Changing Operating System Colors

Internet Explorer respects operating system color changes, such as those made by high contrast mode in Windows. This allows users to create color schemes that provide greater contrast and make forms more readable as a result.

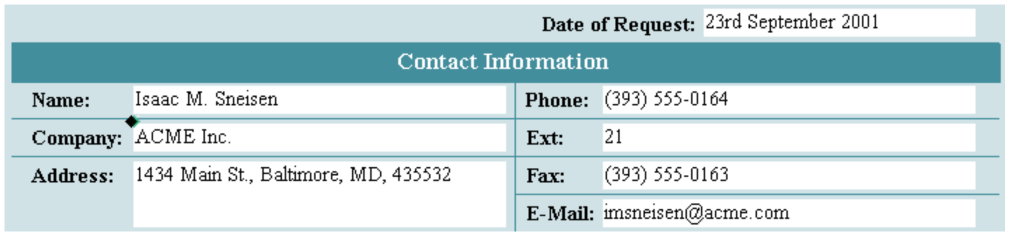
Aids to Navigating Forms

Webform Server includes the following accessibility features to assist users with disabilities when navigating through forms:

- “The Document Accessibility Wizard” on page 68
- “The Focus Indicator”
- “Accessing Toolbar Buttons” on page 68
- “Keyboard Shortcuts” on page 68

The Focus Indicator

Webform Server offers a special *focus indicator* that emphasizes which form item currently has the input focus. In the following diagram, the focus indicator appears next to the ″Company″ field, showing that it has the focus:



		Date of Request:	23rd September 2001
Contact Information			
Name:	Isaac M. Sneisen	**Phone:**	(393) 555-0164
Company:	ACME Inc.	**Ext:**	21
Address:	1434 Main St., Baltimore, MD, 435532	**Fax:**	(393) 555-0163
		E-Mail:	imsneisen@acme.com

To active the focus indicator, click **Accessibility Mode** in the Webform Server toolbar or type **CTRL + H**.

The Document Accessibility Wizard

Webform Server offers a special *document accessibility wizard*. When activated, the document accessibility wizard divides up the form so that each user input item is displayed on its own page. This allows users to page from item to item, following the tab order provided by the form. Furthermore, this allows users to set high zoom levels without having to constantly scroll to view the full page.

This functionality meets the standards set by the W3C Web Content Accessibility Guidelines. For more information, see http://www.w3.org/TR/WAI-WEBCONTENT/.

Accessing Toolbar Buttons

Users can access Webform Server toolbar buttons with their keyboard, as shown:

Toolbar Action	Keystrokes
Open Form	**ALT - O**
Save Form	**ALT - S**
Print Form	**ALT - P**
Refresh Form	**ALT - R**
Toggle Accessibility Mode (Focus Indicator)	**ALT - H**
Toggle Wizard Mode (Document Accessibility Wizard)	**ALT - A**
About Webform Server (Info)	**ALT - I**

Keyboard Shortcuts

The following table describes how users can perform basic navigation tasks with the keyboard:

Action	Keystrokes
Move to next focus item	**TAB**
Move to previous focus item	**SHIFT + TAB**
Scroll through list choices while list item has focus	**DOWN ARROW** or **UP ARROW**
Choose list item when highlighted	**ENTER**
Show list choices while the list item has focus	**ALT + DOWN ARROW** or **ALT + UP ARROW**
Select a check box or radio button, or to view a list for a popup or combobox	**SPACE**

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. 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. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks

Program Number: 5724-N08

Printed in USA

G325-2595-00