Contents

Introduction

IBM® Workplace Forms™ Server – Webform Server translates XFDL documents into HTML/JavaScript 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 plug-ins or other programs.

About this document

This document explains how to optimize your Webform Server system architecture for the best possible performance. This configuration will differ between each organization based on differing needs. Some systems will need to serve many users at once, others will have only one or two very large forms, others will have several small forms, and so on. This means that each Webform Server setup will need to be tested and tuned individually. As a result, this document describes optimization techniques in general terms. Using these techniques and practices will improve the performance of your system; however, it is impossible to predict how much effect each practice will have on any given system.

Furthermore, while you can improve the performance of your Webform Server application by optimizing your system architecture, the overall performance of your system relies on the optimization of your forms. For advice on designing high performance forms that will work with Webform Server, refer to the *Workplace Forms Server – Webform Server Best Practices Guide*.

Note: This document does not describe how to install, configure, and administrate Webform Server. For this information, see the *Workplace Forms Server – Webform Server Administration Manual*.

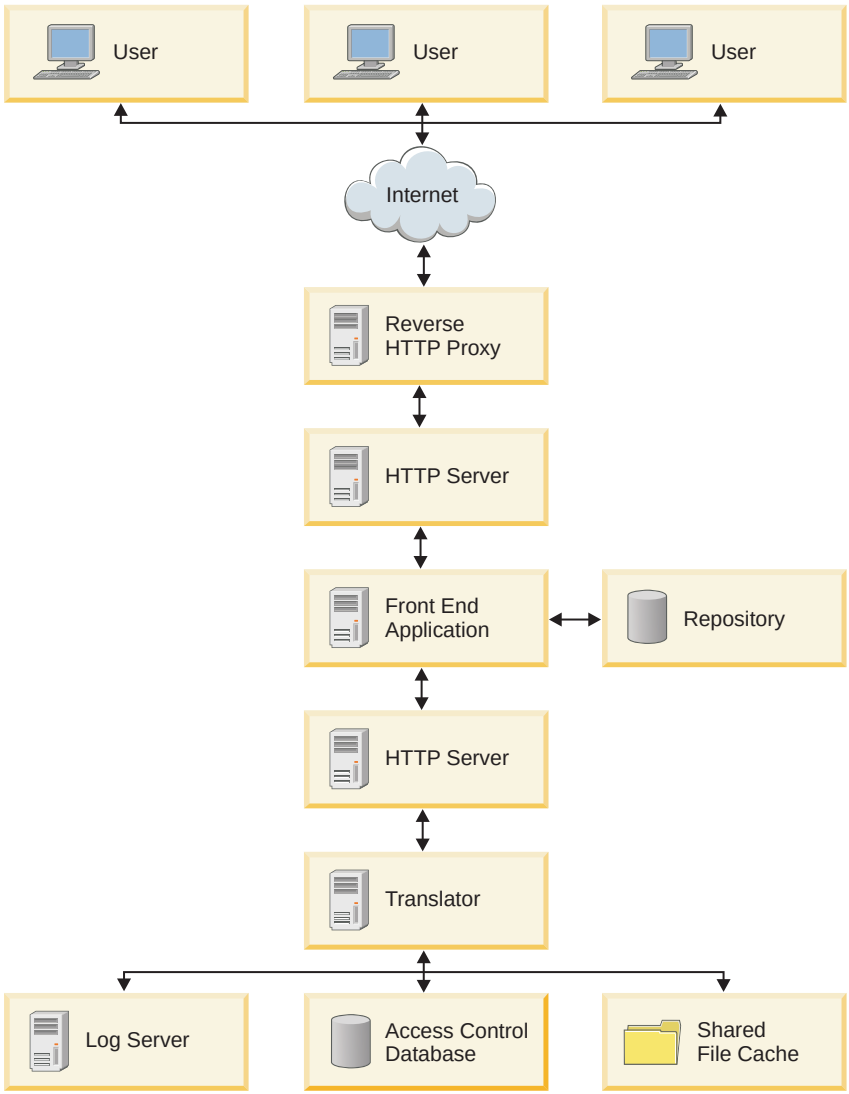
Who should read this document

This document is intended for system administrators and architects responsible for designing, installing, configuring, or maintaining Webform Server components. This document assumes you are familiar with basic system administration and form, application, and system development.

Architecture overview

The following diagram represents a suggested Webform Server system architecture for high performance.

A Basic Webform Server System



Webform Server system components

A Webform Server system is a collection of hardware and software components that includes the use of Webform Server for delivering forms to end-users.

A typical Webform Server system consists of the following primary components:

- Front End Application
- Repository
- Translator

- Shared File Cache
- Access Control System
- Log Server
- HTTP Server

Front End Application

The Front End Application controls communication between end-users, the Translator, and possibly other applications (for example, a form repository, workflow application, and so on). You can also use the Front End Application to populate a form with data before presenting the form to a user, and to extract data from a form after the user submits it.

You create the Front End Application by doing one of the following:

- Create a servlet and deploy it within an application server (IBM WebSphere® Application Server). Webform Server includes a servlet sample that represents a simple Front End Application.
- Create a portlet and deploy it within a Web portal (IBM WebSphere Portal). Webform Server includes a portlet sample that represents a simple Front End Application.

You also use the Workplace Forms Server – API for populating forms with data, validating submitted forms, extracting data from submitted forms, and performing other form processing functions.

Repository

The Repository is a database, file system, content management system, or other back end system for storing forms and associated files.

The repository may also:

- Provide the Front End Application with data for pre-populating a form
- Store or use the data submitted by users and extracted from the form by the Front End Application

Translator

The Translator converts forms between XFDL and HTML and stores form instances (forms that an end-user is interacting with) in memory within its form cache.

The Translator consists of the following:

- Translator servlet
- IBM WebSphere Application Server
- Workplace Forms Server – API

Shared File Cache

The Translator stores form instances (forms that an end-user is interacting with) in memory within its form cache and on disk within the Shared File Cache.

The Translator's form cache can store a specific number of form instances in memory. When this number is exceeded (for example, when many users are simultaneously completing forms), then some form instances are saved to the Shared File Cache.

The Shared File Cache is shared between multiple instances of the Translator, ensuring high availability. If one Translator instance or server fails, another instance or server can continue the user session by accessing the data from the Shared File Cache.

Access Control System

The Access Control System tracks form instances in the Shared File Cache. The Access Control System also keeps track of which forms instances are in use, when each form instance was last accessed, and which user session is associated with each form instance.

You can deploy the Access Control System using one of two methods:

- Access Control Server – Uses a file system on a single server.
- Access Control Database – Uses an SQL database.

Log Server

The Log Server logs activity for the Translator and the Front End Application servlet/portlet.

HTTP Server

HTTP servers are used within high performance Webform Server systems to:

- Handle HTTP connections between the reverse proxy server and the Front End Application
- Handle HTTP connections between the Front End Application and the Translator
- Provide software load balancing across a cluster of Front End Application servers (see "Deploying the Front End Application" on page 9)
- Provide software load balancing across a cluster of Translator servers (see "Deploying the Translator" on page 14)
- Provide SSL secure connections

For high performance Webform Server systems, use IBM HTTP Server as the HTTP server.

For simple Webform Server systems that do not involve a cluster of Front End Application servers or a cluster of Translator servers and do not require high performance, you can use the internal HTTP server functionality provided by IBM WebSphere Application Server instead of dedicated HTTP server software.

Hardware load balancer

Hardware load balancers distribute work between a cluster of servers. Load balancing can also be performed by software, or by a combination of hardware and software.

SSL accelerator

Hardware SSL accelerators perform the encryption algorithms involved in SSL translations.

Recommendations for initial setup and tuning

Use the recommendations in this section to setup your intial Webform Server system and to tune it for maximum performance.

Setting up your overall system

Consider the following when setting up your overall system.

Setting up servers and virtual servers

You can configure a single multi-processor server into several *virtual servers*. All virtual servers on a single physical server share the same overall hardware, but each virtual server behaves as a separate machine.

For example, you may have one physical server that hosts several virtual servers. You then use each virtual server to host a different Webform Server component.

Support for virtual servers differs for each hardware vendor and operating system. For example, IBM's Logical Partitioning (LPAR) is a virtualization system architecture that supports several operating systems. Refer to your hardware and operating system documentation for detailed information on setting up virtual servers.

Note: You can configure a virtual server as follows:

- All virtual servers on a single physical server share CPUs, physical memory, and disk volumes.
- Each virtual server on a single physical server has its own dedicated CPUs, physical memory, and disk volumes.

When configuring virtual servers for use in a Webform Server system, use dedicated CPUs, physical memory, and disk volumes for each virtual server.

Install each component on a dedicated server

Install each system component on a dedicated server (or virtual server):

- Translator (You can install multiple Translator instances on a single server)
- Log Server
- Access Control System
- Shared File Cache
- HTTP Server
- Front End Application
- Repository

Do not install multiple components onto a single server.

Do not install any unrelated software onto the servers hosting the Translator.

Apply latest patches

Apply the latest patches for all software (operating system, Webform Server, WebSphere Application Server, DB2®, IBM HTTP Server, WebSphere Portal) during initial setup. Check for additional patches as a part of routine system maintenance.

Do not use multiple heaps for memory allocation on AIX

Do not configure AIX® to use multiple heaps for memory allocation on any system running a Java™ application. This includes the servers hosting the Translator, the Front End Application, and the Log Server. This feature causes problems with the Java Virtual Machine (JVM).

Provide the Viewer to some of your users

Whenever you use Webform Server to deploy forms to a group of users, consider providing the Viewer to some of your users.

For example, your user base might include an external group (for example, your customers who are submitting applications) and an internal group (for example, your employees who are approving customer applications). In this situation, you can provide the Viewer to your employees.

If Webform Server detects that the end-user has the Viewer, then the XFDL form is directly served to the user without conversion to HTML. This will help reduce the loading of the Translator and increase overall performance.

See also "Disable Viewer detection if not using Viewer" on page 12.

File Descriptor Limit

During testing, set the maximum number of open file descriptors (UNIX®) or file handles (Windows®) to unlimited. During production, you may want to adjust this setting based on the needs of your application.

A *file descriptor* (UNIX) or *file handle* (Windows) is a number that the operating system assigns temporarily to a file when it is opened and uses internally when accessing the file. By default, the operating system limits the number of file descriptors/handles that each process may open and the total number of file descriptors/handles that all processes together may consume.

On UNIX systems, you set the maximum number of open file descriptors using the **ulimit** command.

On Windows systems, refer to your operating system documentation for detailed information.

Network

Consider the following when selecting and setting up your network:

- Locate all hardware components on the same LAN and ensure that the LAN is 100BaseT or better (a Gigabit Ethernet is preferred).
- Do not deploy hardware components over a WAN or campus area network.
- On Windows systems, tune TCP/IP by setting the close_wait interval to 30 seconds.
- If your application requires increased failover support, you must allow for higher bandwidth between the Translator and the Shared File Cache (for

example, using a higher bandwidth network, using a dedicated LAN and mounting the Translator and Shared File Cache via NFS, or using a SAN).

Designing forms

The performance of your Webform Server system is highly dependent upon the design of the forms being used.

The overall size of your forms and the number of pages, items, and computes that it contains can adversely affect the performance of Webform Server applications. To ensure the best performance, architects of Webform Server solutions should work closely with form designers to ensure that your system is tuned to support your forms. Furthermore, form designers should review "Best Practices" and "What's New" documentation to ensure that they are familiar with form design best practices, new functionality, and functionality changes.

In general, you should use XForms-type forms when integrating forms with applications that already use XML, especially if those applications offer XML interfaces. XForms allows you to easily extract XML instance data from the form instead of creating a custom module to extract it. Furthermore, XForms allows you to format the data to match any schema and validate the data against the schema before submission.

Even if your application does not use XML, XForms can still benefit your system. The data model simplifies copying information from one page to another, making wizard-style forms easier to create and manage.

Additionally, forms for high performance applications should consist of five or fewer pages, with each page containing less than 30 items. From a performance perspective, if you need to collect a lot of information it is better to provide a series of smaller forms than to provide one large form. An application that uses five forms with five pages each will outperform an application that uses one form of 25 pages. The form series can still share data, including form pre-population, by passing the data model (or parts of the data model) from one form to another.

For detailed information on designing forms for use in a high performance Webform Server application, see the *Webform Server Best Practices Guide*.

Compressing forms

In general, do not compress your forms. Compressed forms significantly increase the CPU loading on the Translator server.

Using uncompressed forms will result in higher data transfers over your network. If your forms are very large, network bandwidth may become an issue. In these situations, you may prefer to compress your forms to reduce network loading.

Setting up the Front End Application

Consider the following when setting up and tuning the Front End Application.

Deploying the Front End Application

You can deploy the Front End Application as follows:

- Deploying a single instance of the Front End Application on a single server may be acceptable for some applications.

- Deploying multiple instances of the Front End Application across a cluster of servers provides failover support and allows for load balancing and may be required for high performance applications.

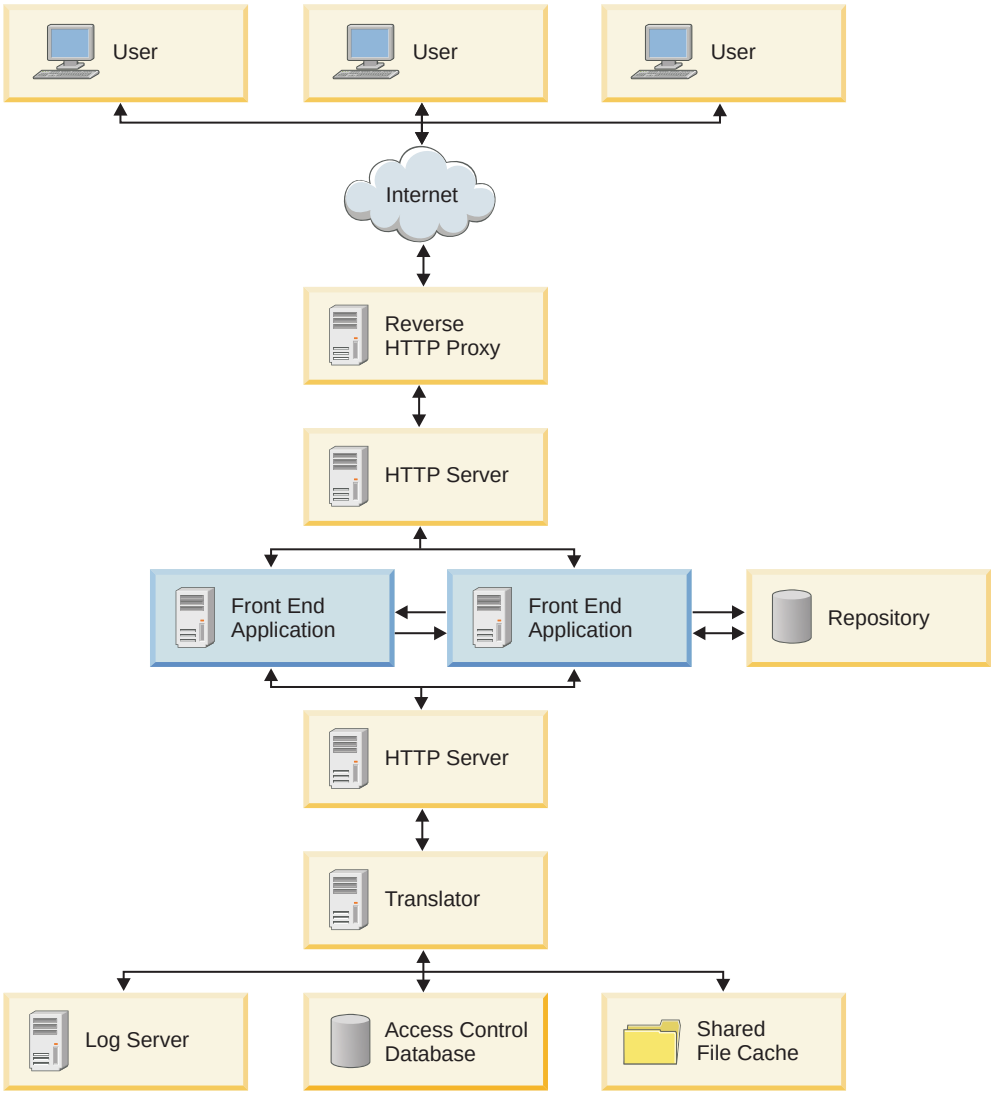
Start with one server hosting the Front End Application. If testing reveals that the server is CPU-bound, you may want to deploy an additional server hosting the Front End Application to increase overall performance.

If your Front End Application is performing a large amount of processing, then you may want to use more servers.

When deploying the Front End Application across a cluster of servers, you must:

- Provide load balancing between each instance of the Front End Application (using IBM HTTP Server or a hardware load balancer).
- Configure WebSphere Application Server for clustering.

Deploying the Front End Application Across a Cluster of Servers



Complementary design of Front End Application and forms

When the Front End Application presents the user with an HTML page, that page contains one or a combination of the following:

- HTML and JavaScript™ generated by the Translator from an XFDL form
- HTML generated by the Front End Application (for example, using servlets, JSF, and so on)

Design your forms and your Front End Application so that HTML content is generated using the appropriate method. You can optimize overall performance by generating more content with the Front End Application and generating less content with the Translator.

For example, use the Front End Application to:

- Present the user with a login screen.
- Present a list of forms that the user can select.
- Present a confirmation message after form submission.
- Present a summary page containing data extracted from the form.

Use an XFDL form to:

- Collect user data.
- Provide a precise layout of items.
- Provide highly controlled navigation within the form.

Configure the Front End Application for minimal logging

Front End Application servlet/portlet events are logged to the Log Server. If you configure the servlet/portlet for increased logging, the performance of the Log Server and the overall Webform Server system will decrease.

When creating the web.xml file for the servlet/portlet, set the logLevel to 0 (only errors and exceptions are logged). If you set logLevel to 1, request URIs and response events are also logged.

Note: You must also configure the Log Server separately. See "Setting up the Log Server" on page 21.

Do not use system garbage collection

Webform Server applications often rely on the Workplace Forms Server – API. However, the default method of system garbage collection hinders the performance of the Java API. As a result, you should consider using a new method of garbage collection to improve memory management and Webform Server performance.

Starting in Version 2.7 of Webform Server, a new memory management model is available for the Classic API which mimics the memory management system used by the Streaming API. You can now determine whether the Java API should request system garbage collection (GC) when destroying a FormNodeP object. If system GC is used, all running processes are suspended while the garbage collection takes place. If system GC is not required, objects notify the Java API when references to the object are no longer needed. For most objects, this notification is automatic. This allows for significant performance improvements as unnecessary system GC requests are eliminated.

To take advantage of this new memory management model, you must set the setHardGCFlag method to **false**, which disables system garbage collection. For more information about setHardGCFlag and garbage collection, see the technote at http://www-1.ibm.com/support/docview.wss?rs=2357&contex.

Form data population and extraction

For forms that contain a data model (XForms or XML Model), do not populate data or extract data for individual elements:

- When populating a form with data for all elements within an instance, populate the entire instance at the root level. Populating elements individually is much slower.
- When populating a form with data for all child elements of a parent, populate the parent element at the sub-root level. Populating child elements individually is much slower.
- When extracting data from a form, extract the entire instance. Extracting elements individually is much slower.

Disable Viewer detection if not using Viewer

By default, the Front End Application will detect if the end-user has the Viewer installed on their system. If you know that none of your users have the Viewer, you can improve performance by disabling automatic detection.

Add the following to the Front End Application's web.xml file:

```
<init-param>
   <param-name>disableViewerDetection</param-name>
   <param-value>true</param-value>
</init-param>
```

The disableViewerDetection setting is not a documented feature.

See also "Provide the Viewer to some of your users" on page 8.

Use the Streaming API

If you are developing a Front End Application using the Workplace Forms Server - API, use the Streaming API (new in Workplace Forms Server 2.7) unless your application requires functions provided only in the Classic API. The Streaming API uses less memory than the Classic API. The performance of the Streaming API depends on the form and your use of the Streaming API.

When using the Streaming API, you can improve performance by using the addHint and addHints methods. These methods let you tell the parser which nodes will be referenced by your application or method, allowing the parser to collect all the requested data while traversing the form once.

Use Server Speed Flags

If you are developing a Front End Application using the Workplace Forms Server - API, you can significantly improve performance by turning on the server speed flag setting in the readForm method.

When a form is read into memory, it evaluates the form data. This includes evaluating computes, detecting duplicate sids, formatting, XForms processing, and so on. As a result, it takes longer to read the form into memory. However, your application may not always need these evaluations to take place when the form is

read. In fact, if you are using the Streaming API, form data evaluation is not permitted when reading the form into memory. However, if you are using the Classic API, the readform method can be configured to only perform specified evaluation behaviors. This is done through the **flags** parameter. The **flags** parameter has the following settings:

0 No special behavior.

XFDL.UFL_SERVER_SPEED_FLAGS
Turns off the following features: computes, automatic formatting, duplicate sid detection, XForms processing, write relevant, the event model, and relative page and item tags (for example, *itemprevious*, *itemnext*, and so on). It also ignores errors. As a result, this setting significantly improves server processing times.

Note: This setting does not respect XForms relevance.

XFDL.UFL_AUTOCOMPUTE_OFF
Reads the form into memory, but disables the compute system so that no computes are evaluated.

XFDL.UFL_AUTOCREATE_CONTROLLED_OFF
Reads the form into memory, but disables the creation of all options that are maintained only in memory (for example, *itemnext*, *itemprevious*, *pagenext*, *pageprevious*, and so on).

XFDL.UFL_AUTOCREATE_FORMATS_OFF
Reads the form into memory, but disables the evaluation of all format options.

XFDL.UFL_XFORMS_INITIALIZE_ONLY
Turns off the following features: controlled item construction, UI connection to the XForms model, action handling set up, and the rebuild/recalculate/revalidate/refresh sequence after instance replacements. This flag respects XForms relevance and validity settings.

XFDL.UFL_XFORMS_OFF
Turns off XForms processing, including UI connection to the XForms model. The primary use of XFDL.UFL_XFORMS_OFF is to turn XForms processing *on* in XFDL.UFL_SERVER_SPEED_FLAGS. This is done by negating XFDL.UFL_XFORMS_OFF with a bitwise NOT and including it with the XFDL.UFL_SERVER_SPEED_FLAGS setting with a bitwise AND. For example:

```
(XFDL.UFL_SERVER_SPEED_FLAGS &(~XFDL.UFL_XFORMS_OFF))
```

The fastest setting is XFDL.UFL_SERVER_SPEED_FLAGS, as it performs the fewest evaluations. The slowest setting is 0, as it performs all of the evaluations. If you need some behaviors and not others, you can use multiple flags by combining them using a bitwise OR, AND, or NOT. For example, the following sample disables the evaluation of computes and format options:

```
XFDL.UFL_AUTOCOMPUTE_OFF | XFDL.UFL_AUTOCREATE_FORMATS_OFF
```

For more information about the readform method and how it is used, see the *Workplace Forms Server API – Java API User's Manual*.

Example

The following example demonstrates the use of readForm to load a form into memory without performing any evaluations:

```
private static FormNodeP loadForm() throws Exception
{
XFDL theXFDL;
formNodeP theForm;
   if ((theXFDL = IFSSingleton.getXFDL()) == null)
      throw new Exception("Could not find interface");
   if ((theForm = theXFDL.readForm("formSample.xfd", XFDL.UFL_SERVER_SPEED_FLAGS))
      == null)
      throw new Exception("Could not load form.");
   return(theForm);
}
```

Setting up the Access Control System

Consider the following when setting up and tuning the Access Control System.

Use the Access Control Database

Use the Access Control Database. Do not use the Access Control Server. The Access Control Database performs faster than the Access Control Server. In addition, the Access Control Database provides failover support.

Setup a separate table for each virtual server

For high performance applications that do not require failover support, configure the Access Control Database so each virtual server hosting Translator instances has its own table in the database. You must also configure the Translator for use with those tables and setup a separate Shared File Cache for each virtual server.

See “Deploying the Translator” for more information.

Tune the Access Control Database

Tune the Access Control Database (DB2) for maximum performance. If you do not tune DB2, then overall performance may be impacted. See the DB2 documentation for information on tuning DB2.

Setting up the Translator

Consider the following when setting up and tuning the Translator.

Deploying the Translator

You can deploy the Translator in several ways:

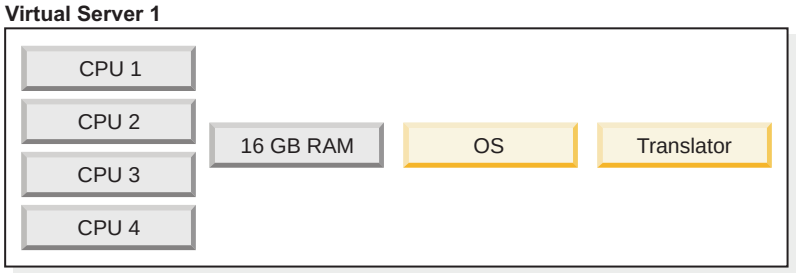
- Deploying a single instance of the Translator on a single server (or virtual server) may be acceptable for very simple applications that do not require high performance or failover support.
- Deploying multiple instances of the Translator on a single server (or virtual server) provides some failover support and is required for high performance applications. (If one Translator instance fails, then another Translator instance can continue the user session.) Increasing the number of Translator instances on a server is referred to as *vertical scaling*.
- Deploying multiple instances of the Translator across a cluster of servers (or virtual servers) provides increased failover support and may be required for some high performance applications. (If one Translator server fails, a Translator instance on another server can continue the user session.) Increasing the number of servers hosting Translator instances is referred to as *horizontal scaling*. If you deploy the Translator in this manner, you must provide load balancing between

the servers (for example, using IBM HTTP Server). You must also configure each instance of WebSphere Application Server for clustering.

Use the following steps to help you setup, test and tune your Translator configuration:

1. For your initial setup, start with one virtual server hosting one instance of the Translator.

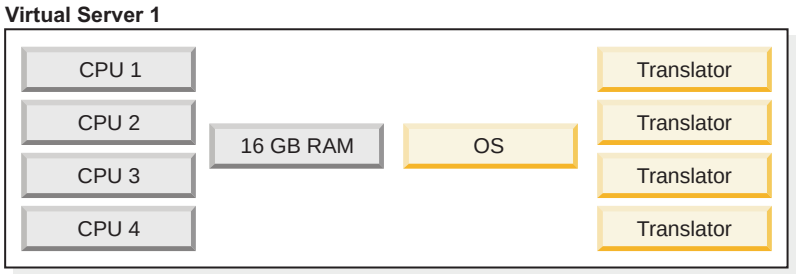
   Virtual Server with 1 Translator Instance



2. After initial testing, adjust the Translator server configuration by increasing the number of Translator instances on the virtual server until the CPU or memory becomes 100% utilized.

   Note: Make sure each Translator instance has no less than 4 GB of physical memory available. For example, if your virtual server has 16 GB of memory, you can increase the number of Translator instances to a maximum of 4.

   Virtual Server with 4 Translator Instances



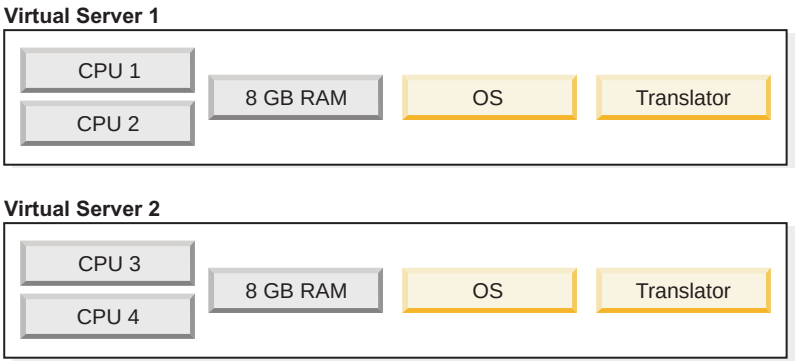
   In general, the virtual server will become memory-bound (the memory is the bottleneck) for larger forms or CPU-bound (the CPU is the bottleneck) for smaller forms.

   Occasionally, the virtual server will become I/O bound, especially when configured to provide better failover support, which requires frequent reading from and writing to the Shared File Cache.

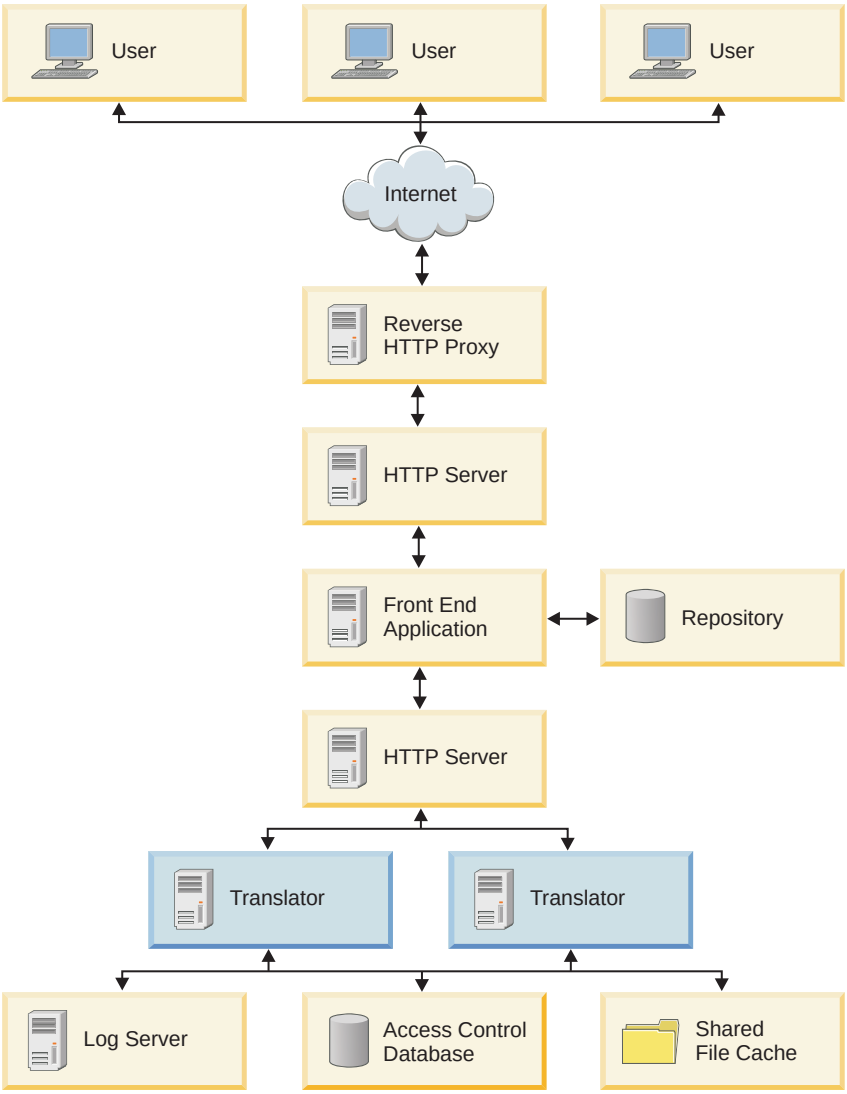
3. Next, re-partition the server into a greater number of virtual servers. For example, if your server has four CPUs, your inital configuration would be 1 virtual server (with 4 CPUs) and your next configuration would be 2 virtual servers (with 2 CPUs each).

   Note: High performance Webform Server applications require at least 2 CPUs per virtual server hosting Translator instances.

2 Virtual Servers with 1 Translator Instance per Server



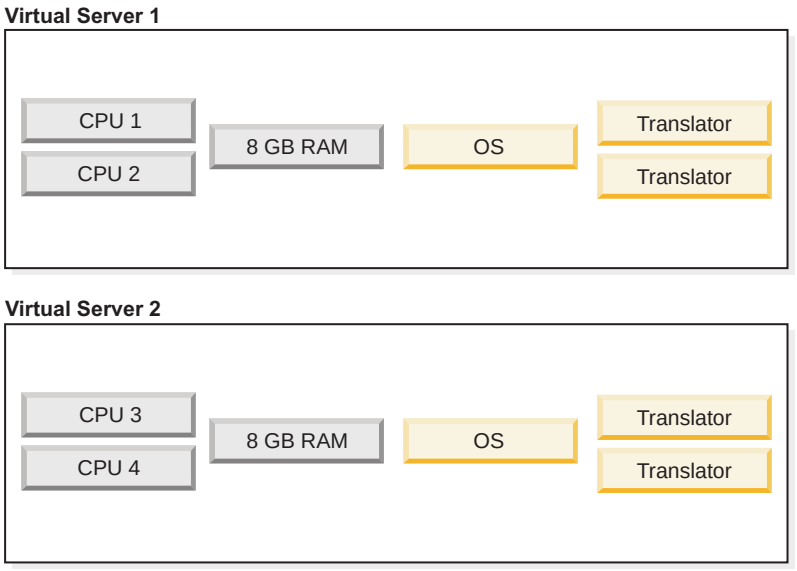
Deploying the Translator Across a Cluster of Servers



4. Experiment with different combinations of virtual servers and Translator instances until resources are fully utilized.

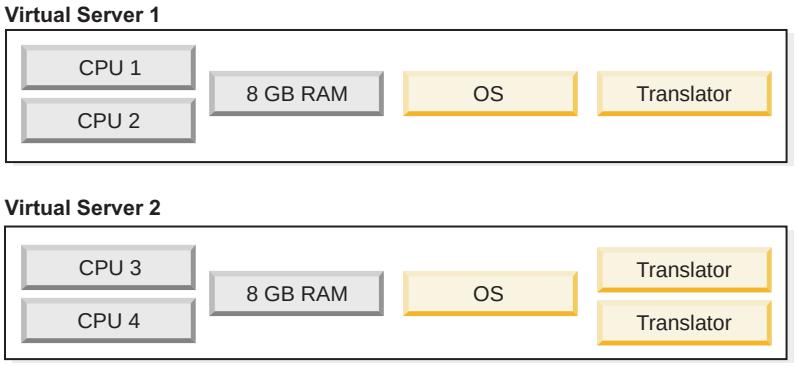
Note: Your Webform Server application will become more complex as you increase the number of virtual servers and Translator instances. Keep in mind the management of your overall system.

2 Virtual Servers with 2 Translator Instances per Server



5. If your Webform Server application consists of several forms of various sizes, consider hosting smaller forms on a dedicated virtual server and larger forms on a dedicated virtual server. Performance is highly dependent upon form size and design, so this will allow you tune each virtual server in different ways. For example, each virtual server could have a different number of Translator instances.

2 Virtual Server Configurations



6. If testing and tuning does not result in acceptable performance, adjust your configuration as follows:
   - If virtual servers are memory-bound, increase the amount of memory for the virtual server by adding physical memory to the server. Continue increasing the amount of physical memory until memory utilization falls just below 100 percent.
   - If virtual servers are CPU-bound, add more CPUs to the server or deploy an additional server for hosting Translator instances.

Setting the process size for the Translator server

Configure the Translator server operating system to use the maximum allowed process size.

Regardless of how much physical memory a server has, the amount of memory that each process can address is limited by the operating system. On a 32-bit operating system, each process can address 4 GB of memory. By default, the operating system reserves some of the 4 GB for kernel usage and some for application usage. However, you can configure some operating systems so more memory is reserved for application usage and less for kernel usage.

The memory available for application usage is the amount of memory available for each Translator instance's form cache. You must set the size of the form cache based on the available memory. See "fcCacheSize."

- **AIX** – By default, AIX (32 bit) reserves 1.75 GB for kernel usage and 2.25 GB for application usage. Using the very large address-space model, AIX reserves 0.75 GB for kernel usage and 3.25 GB for application usage. This results in a very slight decrease in operating system performance, but will allow you to increase the number of simultaneous users that can access the application at maximum performance.
- **Linux®** – By default, many versions of Linux (32 bit) reserve 1 GB for kernel usage and 3 GB for application usage. Refer to your operating system documentation for information on configuring these values.
- **Solaris** – By default, Solaris (32 bit) reserves 0.25 GB for kernel usage and 3.75 GB for application usage. Refer to your operating system documentation for information on configuring these values.
- **Windows** – By default, Windows reserves 2 GB for kernel usage and 2 GB for application usage. On some versions of Windows, using the /3GB option reserves 1 GB for kernel usage and 3 GB for application usage.

Configure the Translator for high performance

Consider the following when configuring the Translator.

fcCacheSize

Configure each Translator instance so its in-memory form cache can make full use of the memory available for each process.

You set the form cache size for a Translator instance using the fcCacheSize option, which represents the maximum number of forms kept in the memory cache. (This value also represents the number of users that can simultaneously access the Translator instance at maximum performance.)

If there is 2.25 GB of memory available for each process, and your form is 10 MB *in memory*, then setting fcCacheSize to 200 will result in 200 x 10 MB = 2 GB (which will use most of the available memory).

Note: The size of a form in memory is larger than on disk. For example, a form that is 1 MB on disk may be 10 MB in memory. One way you can approximate the size of a form in-memory is to load the form in the Viewer, fill in all the form data and attach any files, and monitor the process memory usage.

If your Webform Server system consists of 8 Translator instances, then your system can support 8 x 200 = 1600 simultaneous users at maximum performance. (If more than 1600 users try to use your system simultaneously, then the Translator will write some form instances to the Shared File Cache, and performance will decrease.)

If you set the form cache size too small, then your Webform Server system will support fewer simultaneous users at maximum performance. For example, if there is 2.25 GB of available memory, and your form is 10 MB in memory, then setting fcCacheSize to 100 will result in 100 x 10 MB = 1 GB of memory for the form cache (which will use only half of the available memory).

If you set the form cache size too large, then the Translator will try to make use of more memory than is available, causing the Translator JVM to exit with an Out of Memory error. For example, if there is 2.25 GB of available memory, and your form is 10 MB in memory, then setting fcCacheSize to 300 will result in 300 x 10 MB = 3 GB of memory for the form cache (which is more than the available memory).

fcTimerDelay

Set fcTimerDelay to 90 seconds or greater.

fcTimerDelay controls how frequently the Translator checks in-memory forms for updates and writes them to the Shared File Cache. The default value is 10 seconds.

Lower values provide greater failover support, because in-memory forms are written to the Shared File Cache more frequently. (Once a form is written to the Shared File Cache, any Translator instance can take over session interaction.)

Higher values are recommended for high performance applications. However, the higher the value, the more user data will be lost during a failover, degrading the user experience.

contactFrequency

Do not set contactFrequency any lower than the default value: 900 seconds.

contactFrequncy controls how often a form instance (that is, a form an end user is completing) contacts the Access Control System to indicate that the form is still active. The Access Control System maintains the state of the form and clears inactive forms from the Shared File Cache.

authType

By default, the Translator performs IP authentification (by contacting the Access Control System) on each request to prevent session hijacking. IP authentification failure is indicated by an error in the traslator's log:

```
Couldn't authorize for FormInstance: <Form Instance ID>
Authorization failed for user: <hostname or IP address>
```

If this error occurs frequently in your system, it may be due to your network configuration. Certain network configurations do not correctly pass the user's IP address to the Translator. If so, you can disable IP authentication by setting **authType** to **none** in the translator.properties file. Enabling or disabling IP authentification does not significantly affect performance. However, disabling IP authentification will eliminate this misleading error from appearing in the log.

Note: The authType option is not a documented option.

acCleanerInterval

If testing indicates that your application is I/O bound, experiment with higher acCleanerInterval settings.

acCleanerInterval controls how often the Translator checks the Shared File Cache for old form instances that can be deleted. The default value is 180 seconds.

focusNotificationItems

Depending upon your application, you may want to set focusNotificationItems to none to increase overall performance.

By default, the user's cursor position is posted to the Translator. When you set focusNotificationItems to none, the user's cursor position is not posted to the Translator.

Setting up the Shared File Cache

Consider the following when setting up and tuning the Shared File Cache.

Disk system

Disk I/O, even over networked file systems, is not a bottleneck for typical Webform Server systems.

If testing reveals that your application is disk I/O-bound, and you require failover support, use a higher performing disk system (for example, RAID). If you do not require failover support, use the local file system for the Shared File Cache. See "Use the local file system on the Translator server for disk I/O-bound applications"

See the *Performance Benchmarks for Webform Server* document for information on sizing the disk system for the Shared File Cache.

Use the local file system on the Translator server for disk I/O-bound applications

If testing reveals that your application is disk I/O-bound, and your application does not require failover support (which may be acceptable for very small forms), then you can configure each instance of the Translator to use a local, unshared file system instead of the Shared File Cache. Use a dedicated volume or partition for the Shared File Cache on each virtual server. (Multiple Translator instances on the same virtual server can each use a separate Shared File Cache or they can all share the same Shared File Cache.)

If each virtual server contains only one Translator instance, then there will not be any form of failover support. If one Translator instance fails, the load balancer will transfer the connection to another Translator instance (on a different virtual server) and the user's data from the original Translator instance will be lost.

If each virtual server contains more than one Translator instance, and Translator instances on a common virtual server share the same Shared File Cache, then limited failover support will be provided. If one Translator instance fails, the load balancer will transfer the connection to another Translator instance. If both

Translator instances are on the same virtual server, then the user's data will be maintained. However, if the Translator instances are on different virtual servers, then the user's data will be lost.

Setting up the Log Server

Configure the Log Server for minimal logging (for the Translator and the Front End Application servlet/portlet). Turn off the operational log and debug log. Turn on only the error log.

Note: You must also configure the Front End Application for logging. See "Configure the Front End Application for minimal logging" on page 11.

Note: When you disable the operational log, the concept of an audit trail is lost.

Setting up IBM HTTP Server

Consider the following when setting up and tuning IBM HTTP Server.

Use IBM HTTP Server

Use IBM HTTP Server as your system's Web server.

Do not use the internal HTTP server functionality provided by WebSphere Application Server. IBM HTTP Server provides faster handling of HTTP connections, especially under high load.

Tune IBM HTTP Server

You can increase overall performance by tuning the performance of IBM HTTP Server. See the IBM HTTP Server documentation for information on tuning IBM HTTP Server.

Setting up hardware load balancers

The use of hardware load balancers will not increase overall performance. IBM HTTP Server provides load balancing and is usually sufficient. (The Web servers are usually underloaded by Webform Server traffic alone.)

If your infrastructure already involves the use of hardware load balancers, you can use them in front of or in addition to IBM HTTP Server, in front of the Front End Application, or in front of the Translator. If you are using hardware load balancers in front of the Translator, you must configure the hardware load balancers for session affinity.

Setting up WebSphere Portal

There are no special considerations in deploying a Webform Server application within a portal environment. Webform Server can efficiently handle the frequent content requests made by portlets.

Testing Performance

Every Webform Server system architecture will differ based on the needs of the organization. Some systems will need to service many users at once, others will have only one or two very large forms, others will have several small forms, and so on. As a result, each form and Webform Server setup will need to be tested and tuned individually.

Before you can fine tune your Webform Server system architecture, you must understand how your system resources will be used and how your resources will be allocated under a heavy load. Ideally, all components should be near saturation, so that system resources are not wasted. If you want the highest possible performance from your Webform Server application, you will have to "tweak" the system to see which configuration provides the best results for your particular application. We recommend using tools like Rational® Performance Tester to simulate load and test system, application, and form scalibility. Experimentation is the key for capturing the highest possible performance.

There are a number of tools that you can use to capture and analyze data. These include:

Tools for testing AIX

There are a number of tools and UNIX commands that provide performance information. These include:

nmon Monitors and analyzes performance data, including:

- CPU utilization
- Memory use
- Kernel statistics and run queue information
- Disks I/O rates, transfers, and read/write ratios
- Free space on file systems
- Disk adaptors
- Network I/O rates, transfers, and read/write ratios
- Paging space and paging rates
- CPU and AIX specification
- Top processors
- IBM HTTP Web cache
- User-defined disk groups
- Machine details and resources
- Asynchronous I/O
- Workload Manager
- Network File System

- Dynamic LPAR changes for pSeries® p5 and OpenPower™

For more information regarding nmon, see http://www.ibm.com/developerworks/aix/library/au-analyze_.

UNIX ps command

Reports the performance data for each of the currently running processes. This data includes:

- The process ID
- The terminal the process was started from
- The CPU usage of the process
- The process name

The ps command can be modified to provide additional information as well. For more information regarding the ps command, see http://www.unix.org.ua/orelly/unix/upt/ch38_05.htm.

UNIX Top

Reports the top CPU using processes. For more information regarding Top, see http://www.unixtop.org/

UNIX vmstat command

Reports the virtual memory usage data, including:

- Memory
- CPU activity
- Paging
- Block I/O
- Processes

The vmstat command can be modified to provide additional information as well. For more information regarding the vmstat command, see http://www.linuxcommand.org/man_pages/vmstat8.html.

UNIX iostat command

Monitors system input/output and generates two reports: CPU Usage and Device Usage. These reports include the following data:

- Percentage of CPU use when executing at the user level.
- Percentage of CPU use when executing at the kernel level.
- Percentage of time that the CPU was idle while there was an outstanding input/output request.
- Percentage of time that the CPU was idle without an outstanding request.
- Partition name
- Number of transfers per second.
- Amount of data read (in kilobytes per second and blocks per second).
- Amount of data written (in kilobytes per second and blocks per second).
- Total number of blocks and kilobytes read.
- Total number of blocks and kilobytes written.
- The number of read requests merged and issued per second.
- The number of write requests merged and issued per second.
- The number of sectors and kilobytes read per second.
- The number of sectors and kilobytes written per second.
- The average size of requests (in sectors).

- The average queue length of requests.
- The average time for input/output requests to be served (in milliseconds).
- The average service time for input/output requests (in milliseconds).
- Percentage of CPU time where input/output requests occured.

The iostat command can be modified to provide additional information as well. For more information regarding the iostat command, see http://www.linuxcommand.org/man_pages/iostat1.html.

Tools for testing WebSphere Application Server

Use the WebSphere Application Server Tivoli® Performance Viewer to monitor and analyze performance data for WebSphere Application Server (WAS) from the WAS administrative console.

Tivoli Performance Viewer is a performance monitor that is embedded in WebSphere Application Server. It provides performance data on:

- Servlets and JavaServer pages
- Enterprise beans
- Enterprise JavaBeans™ methods
- Server performance
- Connection pools
- Thread pools

For more information regarding Tivoli Performance Viewer, see http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.j.

Tools for testing Windows

The Windows Administrative Tools set includes a Performance tool. The Windows Performance tool contains two components:

System Monitor
Collects performance data on memory, disk, processor, network, and so on.

Performance Logs and Alerts
Records performance data and sets system notifications to alert you if values change beyond a given range.

You can access Windows Administrative Tools through the Start menu. Go to **Start → Settings → Control Panel**. When the Control Panel opens, double-click Administrative Tools, and then Performance.

For more information about the Windows Performance tool, see http://www.microsoft.com/windows/windows2000/en/advan.

Tools for testing Java

There are a number of tools you can use to ensure that your Java is efficient.

These tools include:

Thread dumps
Allow you to analyze process exceptions with Java applications. For more

information, see *Troubleshooting Java code on AIX: Data collection for AIX core dumps* at http://www.ibm.com/developerworks/aix/library/au-javaonaix_core.html?ca=drs-.

Code profiler
Allows you to pinpoint sections of inefficient code.

Rational Application Developer
Allows you to rapidly design, develop, assemble, test, profile and deploy high quality Java/J2EE, Portal, Web, Web services and SOA applications. It is integrated and optimized for WebSphere Application Server and WebSphere Portal Server and includes test environments for these products.

Tools for testing DB2

If you are using an access control database, you should tune your DB2 system design, database design, and application design.

For more information about tuning DB2, see *DB2 UDB V7.1 Performance Tuning Guide* at http://www.redbooks.ibm.com/abstracts/sg246012.html.

Appendix. Resources

Tuning Webform Server involves tuning the Webform Server forms and application, as well as its hardware and software components, such as WebSphere Application Server, AIX, DB2, and IHS. For more detailed information regarding Webform Server components, see:

- *Workplace Forms – Form Design Best Practices Guide* (Viewer and Webform Server form design principles) at http://publibfp.boulder.ibm.com/epubs/pdf/c2359500.pdf
- *Workplace Forms Server – Webform Server Best Practices Guide* (Webform Server-only form design principles) at http://publibfp.boulder.ibm.com/epubs/pdf/c2359380.pdf or contact the lab for an updated version with additional performance guidelines.
- *Workplace Forms Server – Administration Manual* at http://publibfp.boulder.ibm.com/epubs/pdf/c2359370.pdf.
- *WebSphere Performance Tuning Guide* at http://www.redbooks.ibm.com/abstracts/sg245657.html
- *Troubleshooting Guide for WebSphere Application Server* at http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27005324
- *AIX 5L™ Practical Performance Tools and Tuning Guide* at http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246478.html?Open.
- *Problem Solving and Troubleshooting in AIX 5L* at http://www.redbooks.ibm.com/abstracts/sg245496.html.
- *DB2 UDB V7.1 Performance Tuning Guide* at http://www.redbooks.ibm.com/abstracts/sg246012.html.
- DB2 *Troubleshooting Guide* at http://webdocs.caspur.it/ibm_doc/udb-6.1/db2p0/index.htm
- *IBM HTTP Server Performance* at http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzahx/rzahxebushttp.htm.
- *Troubleshooting Java code on AIX: Data collection for AIX core dumps* at http://www.ibm.com/developerworks/aix/library/au-javaonaix_core.html?ca=drs-.
- *Rational Application Developer V6 Programming Guide* at http://www.redbooks.ibm.com/abstracts/sg246449.html.

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.

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

IBM Corporation
Office 4360
One Rogers Street
Cambridge, MA 02142
U.S.A.

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
AIX
AIX 5L
DB2
OpenPower
pSeries
Rational
Tivoli
WebSphere
Workplace Forms

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Program Number: 5724-N08

Printed in USA