INTERWOVEN

Interwoven® Turbo Product Guide

v2.0

for IBM® WebSphere® Application Server

INTERWOVEN

Interwoven, Inc.

1195 West Fremont Ave.

Sunnyvale, CA 94087

http://www.interwoven.com

Printed in the United States of America

Version 2.0

Part # 40-10-40-42-04-200-600

Table of Contents

Chapter 1

About This Product

What is an Interwoven Turbo Product?

Interwoven® Turbo is a solution software product that enables and accelerates the integration of Interwoven TeamSite® with a number of eBusiness suites. Each Turbo product provides a standardized set of integration points and baseline capabilities for a specific partner platform. However, Turbo does not restrict the customer to that single platform, but rather makes all of them available. Because of Turbo's modular approach and broad-based availability, TeamSite can be coupled with any of the leading eBusiness suites for which a Turbo has been developed.

The advantages of Interwoven Turbo are not limited to the flexibility and standardization it facilitates; Turbo also builds customer value by increasing Return On Investment (ROI) in the following ways:

- **Accelerates time-to-Web**: Turbo gets you up and running fast by eliminating the guesswork in getting enterprise products working together. Professional service expenditures can be targeted to adapting the solution to best meet the customer's environment rather than figuring out how to get the two products to work together.
- **Extends Investment**: As part of Interwoven's solution software, Turbo is licensed under a two-year subscription that entitles a customer to any Turbo upgrades released during the period.
- **Increases Productivity**: Turbo extends and enhances TeamSite's native capability to support all user types, from business users to Web and application code developers. It provides additional mechanisms for interacting with the eBusiness suite that help all contributors be more effective.
- **Content Reuse**: Interwoven Turbo will support a number of best-of-breed eBusiness suites so that customers can reuse content when they modify or build new applications.
- **Operational Stability**: Turbos, like all Interwoven products, are fully supported by Interwoven's enterprise-class technical support.

Features of the Turbo Product Family

Interwoven Turbo extends the functionality of TeamSite by providing components that seamlessly interface with eBusiness suite applications, such as Web application servers and personalization servers. Through these components, Turbo can provide an expanded set of capabilities to TeamSite users, including:

- Virtualization of dynamic Web applications during development
- TeamSite users can walk through an application in their workarea just like a customer
- Support for personalization servers through templating and metadata capture
- Workflow approval of content prior to submission
- Deployment of content (file system and database) to the production environment
- A simple business-user interface for contributing content to the Web properties

Application Server Component

"Out of the box" TeamSite virtualizes static Web content, which allows an entire Web site to be previewed from within a workarea while still in development. The Turbo application server component takes this one step further and provides the virtualization of dynamic content, such as JSPs, EJBs, ASPs, JHTML and servlets. This feature accelerates Web development by enabling in-context QA of the entire site from a workarea without disrupting the work of other contributors.

Additionally, Turbo usually provides an augmentation to the TeamSite User Interface (UI) to interact with the application server and register application components, and provide a mechanism for switching between "local" and "shared" server instances from within the TeamSite workarea.

Personalization Server Component

Turbo also includes a component for deploying template-entered data and metadata to repositories used by personalization servers and eCommerce applications. The dynamic targeting of specific content to particular users or groups can be based on comparisons of this data to the personalization schema constructed for a given business model. TeamSite supports this through Turbo by mapping metadata and the content personalization database so that content can be delivered using the correct rules. Furthermore, TeamSite stores the personalization rules and thus they can be versioned.

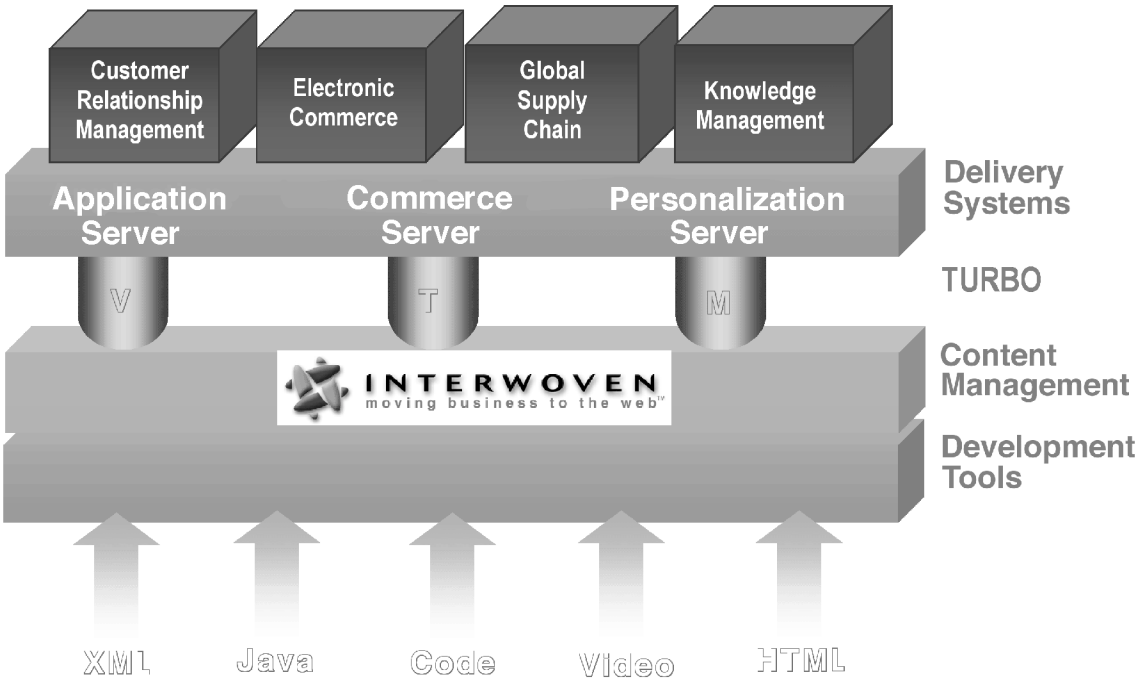
Business-User Support

Business users are the knowledge and process experts within a company. It is crucial that their content contribution be disseminated across the enterprise and effectively published to Web properties. Among the many ways in which Interwoven Turbo facilitates this are providing whole-site preview capability while still in development, updating repositories with content created in TeamSite templating or with specific metadata captured on Web site assets, and enforcing workflow approvals for content submission. Business-user support is a critical function of the Turbo product and will be discussed in greater detail below.

Architectural Overview

General Turbo Architecture

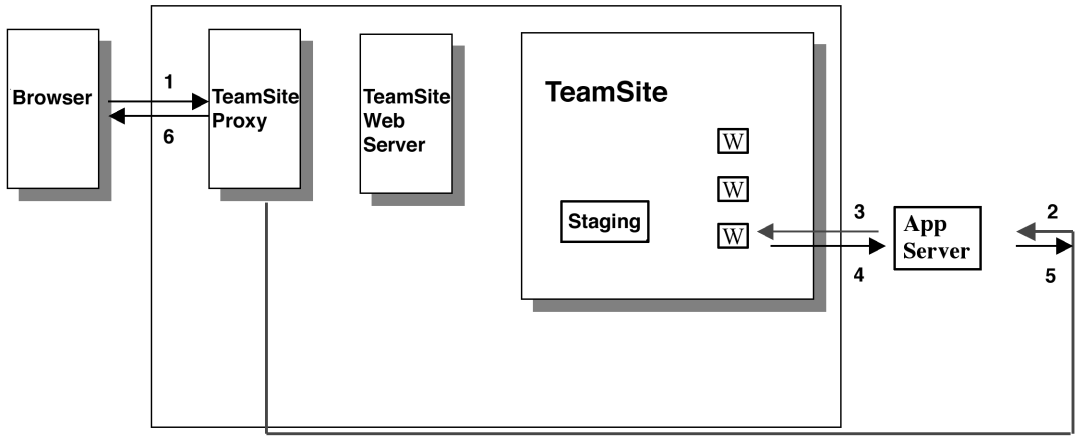
Turbo connects the TeamSite content management functionality to various delivery systems while augmenting the TeamSite native virtualization, templating and metadata capabilities. Development tools and delivery systems are all interconnected by the Turbo architecture. The result is that the customer's development environment is fully vertically integrated, from the initial back-end development of content to its deployment.



Turbo Architecture

Application Servers and Virtualization

The Turbo design is geared toward implementations with application servers. The TeamSite server, proxy server and Web server all interact with the application server to provide virtualization of dynamic, application-driven Web sites, as well as the native capability of TeamSite to virtualize static Web sites. At a high level, this is accomplished by configuring the TeamSite Proxy Server to redirect requests for certain file types to the application server and then pointing the application server toward the correct area of the originating request.

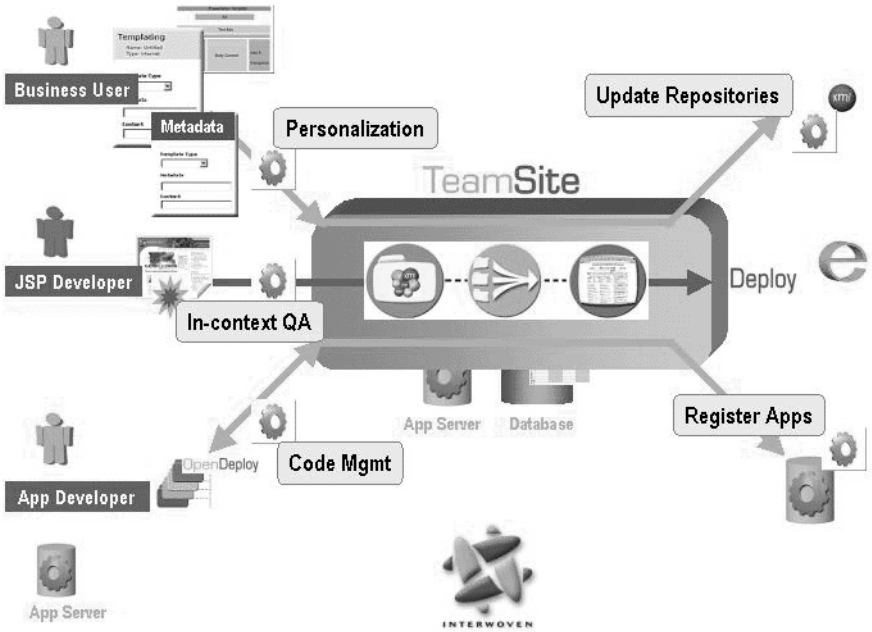


Application Server Integration

Instances of an application server can be set up to correspond either to a single TeamSite workarea or to be shared by multiple workareas, depending upon which types of users are involved. For example, presentation layer developers, such as content contributors and Web page designers, can share a single instance of the application server. On the other hand, Java™ developers should each have a dedicated instance (one application server instance per TeamSite workarea) because their work can affect the behavior of the application server. Turbo provides the flexibility to implement either of these architectures.

Who Can Use This Product?

Interwoven Turbo extends the inherent capability of TeamSite to support multiple types of users. As the diagram below shows, TeamSite with Turbo can connect back-end content contributors and their work to the databases, applications and repositories that power an enterprise-level Web site. The following section will describe three types of contributors—business users, application developers, and presentation-level developers—and discuss how they can benefit from the functionality provided by Turbo.



Interwoven Turbo Users

Business Users

Numerous Interwoven Turbo features let business users add their unique knowledge and expertise to Web properties. These features include templating, structured metadata capture coupled with personalization server support, and workflow approval.

Templating

Templating provides a means for the business user to contribute content through a simple, intuitive interface and then target that content to any number of repositories within the eBusiness platform. Templates are thus a means to capture the intellectual property of the business experts, who can also preview their content before submission. Specifically, the business users can:

- Enter structured content through a data capture template (DCT) based on a schema required by the personalization targeting rules.
- Preview content with presentation template applied to see how it will appear to the end user, and view dynamic content to test personalization rules.
- Publish content to Web properties through a structured and tailored mechanism.

Metadata Capture

Turbo lets business users:

- Apply metadata (tag) Web site assets based on a name/value pair specified in the repository schema and then ensure the deployment of these assets and metadata to the repository.
- Manage the development of and version control of personalization server rules.

Workflow Approval

The use of workflows to approve content before it is submitted provides safeguards and enforces business process.

Application Developers

Application developers develop and maintain the "application" logic and code for an application server. They are primarily concerned with the coding of the business logic and rules and managing access to databases and legacy applications. Turbo helps them through server selection, component registration, and site virtualization.

Turbo benefits application developers with:

- Simplified administration—Full source control for applications using a single TeamSite server.
- Parallel development—Developers can work in multiple workareas creating different pieces of the same application without conflicts.
- Simplified, in-context QA—Developers can preview the entire Web site from within a single workarea.
- Full control—Versioning, workflow, and rollback capabilities provided by TeamSite.
- Distributed development—Local or remote team development.
- Direct access for JSP files—Developers can preview JSP files directly from the TeamSite browser interface.
- Managed deployment of J2EE entities—A UI for managing deployment of J2EE entities to the application server and then for switching between server instances based on the environment.

Presentation Layer Developers

Presentation-layer developers are the content contributors and Web developers who create and modify JSP tags, JavaScript, HTML, DHTML, images and other site elements that will appear at the presentation layer. They are mainly concerned with the layout, design, and functionality of the Web pages. Interwoven Turbo provides support to presentation layer developers by providing in-context QA of dynamic Web sites. This can be done while the site is still in development, thus the development cycle is accelerated.

About This Document

The following Turbo Product Guide discusses the solution and implementation of TeamSite with one of the major eBusiness platform servers. It contains a description of the installation, configuration and implementation of the specific solution that can be used to get the suite component up and running with full TeamSite functionality.

Although this document contains general information about Turbo products that may be similar to other platforms and servers, for detailed information consult the product guide about the specific Turbo that will be implemented.

Note: When describing the installation and configuration of the Turbo product, this document refers to *TeamSite*. If your license is for the *TeamXpress* ™ product instead, the information contained in this document still applies to your product with the possible exception of branch creation. For further details, see the *Interwoven TeamXpress for Multiplatforms,WebSphere Edition Administration Guide*.

Chapter 2

Introduction

This document covers the Turbo product that integrates TeamSite 4.5.1 and IBM® WebSphere® Application Server 3.5.3 and it describes how to install, configure, and use the components that allow TeamSite users to seamlessly interface with the WebSphere Application Server. Installing this software on the TeamSite and WebSphere servers may require `root` access in UNIX, or `Administrator` access in Windows NT® and Windows® 2000.

This guide does not cover the installation or usage of TeamSite or WebSphere Application Server and it is expected that the users and administrators of these software packages are familiar with installing or using them.

Overview

The design of the Turbo product uses a concept called *sandboxes* to give each TeamSite user a virtual copy of a Web site. The administrator creates a pool of sandboxes, and determines pool size based on project needs and amount of resources.

Each sandbox serves as a link from the user's workarea to the WebSphere Application Server. Users check out and check in sandboxes from the pool, and the TeamSite proxy is set so that all URL requests from a user's workarea are remapped to the Application Server sandbox. Working within the sandbox allows each user to make modifications in his workarea without affecting another user's workarea.

Sandboxes

There are two types of sandboxes—*lightweight* and *heavyweight*—each with a different set of capabilities. The WebSphere/Turbo administrator can use these two types of sandboxes to create pools of configurable numbers of sandboxes.

Lightweight Sandboxes

Lightweight sandboxes are intended for users who modify content such as HTML and JSPs. All sandboxes in a lightweight sandbox pool must have the same configuration, as specified by the pool administrator. In a lightweight sandbox, users can modify existing servlets in the configuration, but may not add or delete them, since this would require a reconfiguration of the entire sandbox pool. In WebSphere Application Server, each lightweight sandbox is implemented as a Web application, while the pool is implemented as an application server.

Heavyweight Sandbox

Heavyweight sandboxes allow users to add servlets and EJBs to their configuration, as well as modify content such as HTML and JSPs. Each heavyweight sandbox may have a unique configuration from other sandboxes, lightweight or heavyweight. In the WebSphere Application Server, each heavyweight sandbox is implemented as an application server. As application servers, heavyweight sandboxes allow greater flexibility but require much greater memory resources.

Prerequisite Software Components

The Turbo product requires that you use the following versions of the solution components:

- WebSphere Application Server 3.5.3 (3.5 + fix pack 3)
- One additional eFix for fix pack 3 - PQ46019 (either download from the IBM eFix site or request through IBM support)
- Database (refer to IBM WebSphere Application Server manual for recommendations)
- Web servers for TeamSite (IIS 4.0, NES 3.6.3 or Apache for Solaris™ recommended)
- Web servers for WebSphere Application Server (refer to the IBM WebSphere documentation for recommendations)

Prerequisite Hardware

For TeamSite server configuration requirements, please see the *Administering TeamSite* manual for recommendations.

For hardware recommendations regarding WebSphere Application Server see the corresponding IBM manual.

Chapter 3

Installation and Administration

The installation process consists of two main components, the WebSphere Application Server component and the TeamSite component. The user or administrator installing these components must have file-level permissions to write into the WebSphere and TeamSite directories. TeamSite and WebSphere may run on the same server, or may run on separate servers and interface through Windows File Sharing (Samba) or NFS. If the WebSphere Application Server will be accessing TeamSite files through NFS or Samba, consult the "NFS Configuration" or "Samba Configuration" sections of this document before attempting installation.

Installation of the WebSphere Component

Pre-Installation Procedure

TeamSite/WebSphere administrators need to ensure that the IBM WebSphere Administrative Server is already started and that at least one application server exists. The assumption is that the default server will be available for the installation program to install additional servlet components, or that at least one other application server should exist and already be started. Also TeamSite's `iwserver` share should be mapped on the WebSphere server computer.

Assumptions and Limitations

It is assumed that security is disabled on the WebSphere Administrative Console client, because this Turbo will most likely be used within a development environment with TeamSite rather than in a production environment. This Turbo product will not work if security is turned on because not all configuration file templates contain authentication information.

Extracting and Running the Installation Files for Windows NT

To extract and run the installation files for Windows NT, follow these steps:

1. Log in as `Administrator`, or another user with file write permissions, to the WebSphere Application Server directories.
2. Go to **Start Menu > Control Panel > System > Environment** and set the following system variables if they do not exist:

JAVA_HOME

This variable should point to the root directory of the IBM JDK (for example, `C:\Websphere\Appserver\jdk`).

WAS_HOME

This variable should point to the AppServer directory for WebSphere Application Server (for example, `C:\Websphere\Appserver`).

3. Place the `ibm_turbo.exe` file into a temporary directory and double-click it. This opens the install shield program which will prompt the user to supply installation details.
4. Open a DOS command-prompt window and browse to the location of the directory created during the previous extraction.
5. Change directory to `\Websphere`.
6. Execute the `install.bat` file from the DOS command prompt.
7. Go to the section marked "Installation Configuration."

Extracting and Running the Installation Files for AIX and Solaris

To extract and run the installation files for AIX and Solaris, follow these steps:

1. Log in as `root`, or another user with file write permissions, to the WebSphere Application Server directories.
2. Export the following environment variables:

JAVA_HOME

This variable should point to the root directory of the JDK or JRE (for example, Solaris: **`export JAVA_HOME=/opt/Websphere/Appserver/java`** for sh or ksh).

WAS_HOME

This variable should point to the AppServer directory for WebSphere (for example, Solaris: **`export WAS_HOME=/opt/Websphere/AppServer`** for sh or ksh).

3. Place the `ibm-turbo.tar.gz` file into a temporary directory and extract it using `gunzip` and then `tar`. (for example, **`gunzip ibm_turbo.tar.gz, tar xvf ibm_turbo.tar`**).
4. Change directory to the location of the directory created during the previous extraction.
5. Change directory to `install/Websphere`.
6. Add execute permission to `install.sh` by running the command **`chmod +x install.sh`**.
7. Execute the `install.sh` file (for example, **`./install.sh`**).
8. Go to the section marked "Installation Configuration".

WebSphere Application Server Component Installation and Configuration

The installation program will prompt the user for configuration information. Default values will be enclosed in square brackets [ ] and pressing **Enter** will accept that default value. Note that the installation program is case-sensitive and information must be entered in exactly the form that the program asks for. For example, `y/n?` should be answered with **`y`** or **`n`**, not **`Y`** or **`N`**.

1. For the WebSphere installation path, provide the path to the WebSphere installation (for example, Solaris: `/opt/Websphere/AppServer`).
2. To start installing the WebSphere Application Server component, type **`y`** or press **Enter**. The default value is `y`.
3. The WebSphere Administrative Node Name is the node where the administrative application is located. Get this value from the WebSphere Administrative Client. It will most likely be the computer name.
4. The WebSphere Node Name will most likely be the same name as the Administrative Node Name. The value will default to the Administrative Node Name value.
5. The WebSphere Virtual Host will be the virtual host that hosts the sandbox pool and repository servlets. Find this value in the `Virtual Hosts` section of the WebSphere Admin Client. The default value is `default_host/`.
6. Choose to enable or disable file logging. If enabled, a log file will be created in the *`Websphere_Installation_Path`*`/iwws/logs` directory. The log file will be named with a daily timestamp and will contain information regarding requests made to the sandbox pool for that day. If you choose to disable logging, then no logs will be created. The default value is `logging enabled` or `y`.
7. Enter the location of the TeamSite mount. WebSphere will need to access the TeamSite workareas. If the WebSphere component being installed is located on AIX or Solaris server then the default value will be `/mnt/iwmnt`. If located on a Windows NT server then the default value will be `Y:`. Ensure that `/mnt/iwmnt` is mounted to the TeamSite server's *Virtual Fle System* (`iwserver` on AIX or Solaris, or if on Windows NT, a mapped network drive).

   Note: If TeamSite is installed on Solaris and the WebSphere Application Server is installed on Windows NT, UNC-style resource names must be used for this mount (\\servername\sharedrive). Consult the "Samba Configuration" section below for more information.

8. The Host Address of the WebSphere Application Server will be used by TeamSite to contact the `Repository` servlet. The default value will be the same as the WebSphere Node Name.

9. The Host Port of the WebSphere Application Server is the Web server listening port. The default value is 80.

10. The URL for the Repository Servlet will be `http://hostaddress:hostport/iwws/repository`. The default value will use the information that the user provides for the host address and host port to construct the URL.

11. The Repository Servlet Application Server is the WebSphere Application Server. This server will hold the IWWS Web application that will hold the `Repository` servlet. If this application server does not exist in WebSphere then the installation program will create it. The default value is `Default Server`.

12. The Repository Servlet Servlet-Engine will be housed by the WebSphere Application Server entered in the previous step. If this servlet engine does not exist in WebSphere then the installation program will create it. The default value is `Default Servlet Engine`.

13. Upon successful installation of the `Repository` servlet, verify that it is working by accessing it through a browser at: `http://`*`Websphere_address`*`/iwws/repository`. It should report that a parameter is missing. The appropriate parameters will be supplied later by the TeamSite sandbox component.

14. To start installing the IBM Enterprise Information Portal and Content Manager component of the Turbo product, type **y** or press **Enter** to accept the default value, which is `y`. Refer to the *Interwoven Turbo Product Guide for IBM Enterprise Information Portal and Content Manager* before proceeding with the installation.

15. To start installing the WebSphere Personalization component of the Turbo product, type **y** or press **Enter** to accept the default value, which is `y`. Please refer to the *Interwoven Turbo Product Guide for IBM WebSphere Personalization* before proceeding with the installation.

16. Set up the sandbox pool. See the Sandbox Pool Administration section below for details.

Sandbox Pool Administration

The pool administrator runs the `Sandbox Pool Administration` program to create, delete, allocate, and deallocate sandboxes (Note: command-line options are case sensitive). Before a user may check out (allocate) or check in (deallocate) a sandbox, the pool must be created using the administration program.

On Windows NT systems the administration program is located at
WAS_HOME`\iwws\bin\sandboxAdmin.bat`

On Solaris and AIX systems the administration program is located at
WAS_HOME`/iwws/bin/sandboxAdmin.sh`

Creating a Sandbox Pool

To create a sandbox pool, the following arguments must be provided on the command line:

-**`type heavy|light`** (specifies lightweight or heavyweight sandbox pool)

-**`action create`**

-**`poolSize`** ***`n`*** (where *n* specifies the number of sandboxes in the pool)

-**`configFile`** ***`path`*** (only necessary for the lightweight sandbox pool, where *path* specifies the full path to the XML configuration file)

Here is an example of creating a lightweight sandbox pool containing 10 sandboxes, using an XML configuration that exists in the staging area of a branch named `Website`:

```
sandboxAdmin.bat -type light -action create -poolSize 10 –configFile Y:/
default/main/Website/STAGING/config.xml
```

Here is an example of creating a heavyweight sandbox pool containing three sandboxes:

```
sandboxAdmin.bat -type heavy -action create -poolSize 3
```

Deleting a Sandbox Pool

To delete a sandbox pool, the following arguments must be provided on the command line:

-**type heavy|light** (specifies lightweight or heavyweight sandbox pool)

-**action delete**

Here is an example of deleting a lightweight sandbox pool:

```
sandboxAdmin.bat -type light -action delete
```

Here is an example of deleting a heavyweight sandbox pool:

```
sandboxAdmin.bat -type heavy -action delete
```

Allocating a Sandbox

There may be instances when the administrator will allocate a sandbox from the command line instead of going through the TeamSite GUI. The following command-line arguments are necessary to allocate a sandbox:

-**type heavy|light** (specifies lightweight or heavyweight sandbox pool)

-**action alloc**

-**appName** ***sandbox_name*** (specifies the new name of the sandbox. Note that this name should be in the format of *branch_name/workarea_name* to link it to existing workarea in TeamSite.)

-**docRoot** ***document_path*** (specifies the document root for the sandbox)

-**classPath** ***class_path*** (specifies the classpath for the sandbox, multiple classpaths can be specified and delimited by "**;**" on Windows NT and "**:**" on Solaris/AIX)

-**webPath** ***root_URI*** (specifies the Uniform Resource Identifier—URI—location of the sandbox)

-**configFile** ***XML_config_file*** (only necessary if the type is heavy and the path specified for the location of the XML configuration file must be a full path)

Here is an example of allocating a lightweight sandbox for the workarea named `Andre` under the branch named `Website` with one classpath specified (note that this should be typed on the same line):

```
sandboxAdmin.bat -type light -action alloc -appName Website/Andre

-docRoot Y:/default/main/Website/WORKAREA/Andre

-classPath Y:/default/main/Website/WORKAREA/Andre/java

-webPath /default/main/Website/WORKAREA/Andre
```

Here is an example of allocating a lightweight sandbox for the workarea named `Andre` under the branch named `Website` with multiple classpaths specified. A double-quoted string for the classpath parameter is required in the comand-line arguments to prevent the program from treating "`;`" as a delimiter.

```
SandboxAdmin.bat -type light -action alloc -appName Website/Andre

-docRoot Y:/default/main/Website/WORKAREA/Andre

-classPath "Y:/default/main/Website/WORKAREA/Andre/java;
           Y:/default/main/Website/WORKAREA/Andre/servlets"

-webPath default/main/Website/WORKAREA/Andre
```

Here is an example of allocating a heavyweight sandbox for the workarea named `John` under the branch named `Testing` (note that this should be typed on the same line):

```
sandboxAdmin.bat -type heavy -action alloc -appName Testing/John

-docRoot Y:/default/main/Testing/WORKAREA/John

-classPath Y:/default/main/Testing/WORKAREA/John/java

-webPath /default/main/Testing/WORKAREA/John

-configFile Y:/default/main/Testing/WORKAREA/John/config.xml
```

Deallocating a Sandbox

There may be instances when the administrator will deallocate (check in) a sandbox from the command line instead of going through the TeamSite GUI. The following command-line arguments are necessary to deallocate a sandbox:

-**type heavy|light** (specifies lightweight or heavyweight sandbox pool)

-**action dealloc**

-**appName** ***sandbox_name*** (specifies the new name of the sandbox. Note that this name should be in the format of *branch_name/workarea_name* to link it to an existing workarea in TeamSite.)

Here is an example of deallocating a lightweight sandbox for the workarea named `Andre` under the branch named `Website`:

```
sandboxAdmin.bat -type light -action dealloc -appName Website/Andre
```

Here is an example of deallocating a heavyweight sandbox for the workarea named `John` under the branch named `Testing`:

```
sandboxAdmin.bat -type heavy -action dealloc -appName Testing/John
```

Increasing Sandbox Pool Size

There may be instances when the administrator will need to increase the size of the sandbox pool dynamically without destroying the original pool and then having to recreate the sandbox pool with a new size, because developers may already be using some of the sandboxes in the pool. It is recommended that the system administrator perform this task when there are no active sandbox activities, because there could be a concurrency issue with the `Repsoitory` servlet.

To increase the pool size for a sandbox pool, the following arguments must be provided on the command line:

-type light/heavy (specifies lightweight or heavyweight sandbox)

-action increase

`-poolSize n` (where *n* specifies the number of sandboxes user wants to add)

Here is an example of increasing the pool size from two to five for a lightweight sandbox pool.

Note: This should be typed on the same line and the current size is assumed to be two.

```
sandboxAdmin.bat -type light -action increase -poolSize 3
```

Here is an example of increasing the pool size from two to five for a heavyweight sandbox pool (assuming the current size is two).

```
sandboxAdmin.bat -type heavy -action increase -poolSize 3
```

Installation of the TeamSite Component

PreInstallation Procedure

System administrators need to ensure that the WebSphere Application Server component of the Turbo product is installed on the WebSphere server and that the application server containing the IWWS Web application and the repository is already started. This is because the installation program will set up and create lightweight and heavyweight sandboxes.

Assumptions and Limitations

It is assumed that one branch and one workarea are created prior to this installation program because it will set up and configure a reference implementation example into this workarea. JDK 1.2.2 or above is required for this installation program to run.

Extracting and Running the Installation Files for Windows NT

1. Log in as `Administrator`, or another user that has file write permissions, to the TeamSite directories.
2. Go to **Start Menu > Control Panel > System > Environment** and set the following system variables if they do not exist:

JAVA_HOME

This variable should point to the root directory of the JDK or JRE. (for example, `C:\jdk1.2.2`).

TS_HOME

This variable should point to the home directory of TeamSite (for example, `C:\iw-home`).

3. Place the `ibm-turbo.exe` file into a temporary directory and double-click it. This opens the InstallShield® program which will prompt the user to supply installation details.
4. Open a DOS command-prompt window and browse to the location of the directory created during the previous extraction.
5. Change directory to `\Teamsite`.

6. Execute the `install.bat` file from the DOS command prompt.
7. Refer to the section titled "TeamSite Component Installation and Configuration."

Extracting and Running the Installation Files for AIX and Solaris

1. Log in as `root`, or another user that has file write permissions, to the TeamSite directories.
2. Export the following environment variables:

JAVA_HOME—this variable should point to the root directory of the JDK or JRE (for example, Solaris: **`export JAVA_HOME=/opt/jdk1.2.2`** for sh or ksh).

TS_HOME—this variable should point to the AppServer directory for WebSphere (for example, Solaris: **`export TS_HOME=/opt/TeamSite`** for sh or ksh).

3. Place the `ibm-turbo.tar.gz` file into a temporary directory and extract it using `gunzip` and then `tar` (for example, **`gunzip ibm-turbo.tar.gz`**, **`tar xvf ibm-turbo.tar`**)
4. Change directory to **`install/Teamsite`**.
5. Add execute permission to `install.sh` by running the command **`chmod +x install.sh`**.
6. Execute the `install.sh` file (for example, **`./install.sh`**)
7. Refer to the section titled "TeamSite Component Installation and Configuration."

TeamSite Component Installation and Configuration

The installation program will prompt the user for configuration information. Default values will be enclosed in square brackets [ ] and pressing **Enter** will accept that default value. Note that the installation program is case-sensitive and information must be entered in exactly the form that the program asks for. For example, `y/n?` should be answered with **`y`** or **`n`**, not **`Y`** or **`N`**.

1. To start the TeamSite/WebSphere Application Server component installation program, type **`y`** or press **Enter**. The default value is `y`.

2. Provide the path to your TeamSite installation. The default value is `/iw-home` if the installation is being performed on Solaris and `C:\iw-home` if being performed on Windows NT or Windows 2000 systems.
3. Enter whether WebSphere Application Server runs on a UNIX OS. Because WebSphere and TeamSite can exist on separate platforms, the install program is not able to determine if WebSphere is running on UNIX or Windows NT platforms. This setting is used to provide the correct path separator in the TeamSite GUI. The default value is `yes` or `y`.
4. Enter the host address of the WebSphere server. The TeamSite component will use this address to contact the `Repository` servlet on the specified WebSphere server.
5. Enter the host port of the WebSphere server. The TeamSite component will use this port and the previous address to contact the `Repository` servlet on the specified WebSphere server. The default value is `80`.
6. Enter the URL for the `Repository` servlet. This value is constructed using the host address and host port from the previous steps. It is recommended to use the default value.
7. Enter the location of the `iw.cfg` file. On the Windows NT platform this file is generally located at *`iw-home`*`/etc`, and on the Solaris platform it is generally located at `/etc`.
8. Enter the branch name for the installation program to install the example servlet that can be used to test virtualization functionality.
9. Enter the workarea name for the installation program to install the example servlet.
10. To install the Enterprise Information Portal & Content Manager component of the Turbo product, press **Enter** or type **y** at the prompt. Refer to the *Interwoven Turbo Product Guide for IBM Enterprise Information Portal and Content Manager* before proceeding with the installation.
11. To install the WebSphere Personalization component of the Turbo product, press **Enter** or type **y** at the prompt. Refer to the *Interwoven Turbo Product Guide for IBM WebSphere Personalization* before proceeding with the installation.

Proxy Configuration

The `iwproxy_preconnect_remap` section of the `iw.cfg` file must be configured to redirect HTTP requests to the WebSphere server. A basic example of remapping the requests is included below, but the user should consult the *Administering TeamSite* manual for more information on proxy configuration.

Note: After the `iwproxy_preconnect_remap` section is configured in `iw.cfg`, all requests will go through the WebSphere Application Server except HTML files, GIF images, and DCR data that resides in the `data` directory under `templatedata`. Do not place any files other than DCR data in the `data` directory. To preview any type of Web content, a sandbox will need to be allocated to a workarea for preview purposes.

The following addition needs to be inserted into the `iwproxy_preconnect_remap` section for JSP and servlet requests to any user's workarea, edition or the staging area in any branch to be redirected to the WebSphere Application Server.

```
[iwproxy_preconnect_remap]

   _regex=(^/iw-mount/default/main/(.*)/(((WORKAREA|EDITION)/[^/
   ]+)|STAGING)/(.*\.html)((\?.*)?))$=$1

   _regex=(^/iw-mount/default/main/(.*)/(((WORKAREA|EDITION)/[^/
   ]+)|STAGING)/(.*\.gif)((\?.*)?))$=$1

   _regex=(^/iw-mount/default/main/(.*)/(((WORKAREA|EDITION)/[^/
   ]+)|STAGING)/templatedata/(([^/]+/)+)data/(.*))$=$1

   _regex=/default/main/(.*)/(((WORKAREA|EDITION)/[^/]+)|STAGING)/
   (.*)=http://WebSphereServer/default/main/$1/$2/$5
```

Here is another example that demonstrates how to set up the `iwproxy_preconect_remap` section for JSP and servlet requests to any user's workarea, edition or the staging area in the branch named `Test` to be redirected to the WebSphere Application Server only.

```
   _regex=(^/iw-mount/default/main/Test/(((WORKAREA|EDITION)/[^/
   ]+)|STAGING)/(.*\.html)((\?.*)?))$=$1

   _regex=(^/iw-mount/default/main/Test/(((WORKAREA|EDITION)/[^/
   ]+)|STAGING)/(.*\.gif)((\?.*)?))$=$1
```

```
_regex=(^/iw-mount/default/main/Test/(((WORKAREA|EDITION)/[^/
]+)|STAGING)/templatedata/(([^/]+/)+)data/(.*))$=$1

_regex=/default/main/Test/(((WORKAREA|EDITION)/[^/]+)|STAGING)/
(.*)=http://WebSphereServer/default/main/Test/$1/$4
```

NFS Configuration

NFS can be used to export the TeamSite Virtual Directory system so that the WebSphere Application Server, running on another machine on the network, can mount the TeamSite VFS and use it as if the directory were local. To facilitate this, an NFS *server* must be running on the server with the TeamSite instance and an NFS *client* must be running on the server with the WebSphere Application Server instance. For TeamSite on Solaris, consult the *Administering TeamSite 4.x for Solaris* manual for information on how to export and mount the TeamSite VFS through Samba.

Exporting the TeamSite File System on Windows NT

The following instructions detail how to export (share) the TeamSite VFS using *Intergraph DiskShare version 4*.

1. Install Intergraph DiskShare version 4.
2. Go to **Start > Programs > DiskShare Server > DiskShare Configuration**.
3. Select the **NFS Client Groups** tab and then click on the **Add Group** button. Enter ***was_server*** in the `Group Name` field and click **OK**.
4. Click the **Add Member** button. Enter the IP address of the server containing the WebSphere Application Server instance and click **OK**.
5. Select the **Share Options** tab. Enter the mapped drive in the `Share Name` field (the default mapped drive for TeamSite is `Y:\` but this may have been modified during TeamSite installation). Click the **Share** button.
6. In the NFS Share Permissions dialog box, click the **Add** button, select the *was_server* group and click the **Add** button. Change the Type of Access to **Root** and press the **OK** button. Close and accept the settings for the NFS Share Permissions dialog box by clicking **OK**.
7. Click **OK** on the DiskShare Configuration dialog box to accept the new share settings.

8. Verify that WebSphere has access to these directories by logging in as the WebSphere administrator on the server housing WebSphere, mounting the TeamSite directory, and traversing the directories.
9. Determine whether you can traverse into a workarea that is set up exclusively for a particular user. If you are able to do this, your NFS is set up correctly.

Samba Configuration

Samba can be used as an alternative to NFS for sharing the TeamSite VFS. Samba can use an existing Windows NT Primary Domain Controller (PDC) to provide name and password authentication for accessing Samba shares. This has the following benefits:

- Simple account management and usage because account passwords can be maintained on a single system.
- Windows NT systems can use encrypted passwords to access Samba. This increases system security and removes the need to modify these systems to use clear-text passwords to access Samba shares.

Requirements

- A Windows PDC
- UNIX accounts corresponding to Windows NT users

For example: John Doe's UNIX account = *jdoe*, Password = *12345*, PDC= *sys1*, WINS = *10.20.30.40*. This will enable him to access TeamSite on Solaris with his workarea on TeamSite mapped to WebSphere on a Windows NT machine.

Note: If your UNIX user names are not the same as your Windows NT user names, you will have to configure Samba to map between the different user names. See "How to Map Windows NT User Names to Samba User Names" below for assistance.

Configuration Steps

1. Access the following TeamSite Samba configuration file on Solaris:
   `TeamSite/iw-home/samba/lib/iw.smb.conf`
2. Modify or Add the following lines in the `[global]` section of the `iw.smb.conf` file:

```
# Select a Windows NT PDC for your password server; NetBIOS name
password server = sys1
# Use server level security
security = server
# Use password encryption
encrypt passwords = yes
# WINS Server
wins server = 10.20.30.40
```

3. After you set these options, stop and restart your Samba with the following commands:
   `/etc/init.d/iw.samba stop`
   `/etc/init.d/iw.samba start`

4. During the Turbo installation, you will be asked to provide the mounted directory location. Provide the TeamSite virtual directory using the Universal Naming Convention (UNC). The *server* is the computer on which the shared component is located. The *sharedrive* is the network resource name, usually an exported file system or directory that is being shared.

5. After completing the Turbo installation, authenticate the shared TeamSite directory by accessing it through the DOS command:
   `net use\\server\sharedrive` ***`password`*** **`/user:`*`domain`*`\`*`username`**
   or
   Using the **`Run`** command, enter **`\\server\sharedrive`** and type in the user name and password when prompted.

How to Map Windows NT User Names to Samba User Names

This is a useful feature of Samba when you are using a Windows NT PDC for authentication, but have different naming conventions on the system that is running Samba. It can also be used when you need to map a particular user to a specialized account such as `root` or the account used by your Web server.

1. To set up user name mapping, place the following line in the `[global]` section of the `iw.smb.conf` file:
   `username map = /etc/wherever/you/want/the/username.map`

The `username.map` file will look like the following:

```
# unix_username = username1, username2 @unixgroup …
# Note: @group specifies a Unix group
#
# To map NT Administrator and admin accounts to the Unix root account,
# use the following line:
root = Administrator admin
# To map anyone in the Unix webops group to the httpd account, use the
following line:
httpd = @webops
# To map the Windows NT user John Doe to the Unix user jdoe, use the
following line:
jdoe = "John Doe"
```

2. Restart Samba to make use of the user name mapping. If you are using the version of Samba supplied by Interwoven, use the following:
   `/etc/init.d/iw.samba stop`
   `/etc/init.d/iw.samba start`

Uninstall Procedures for WebSphere and TeamSite

If you need to uninstall WebSphere or TeamSite components, use the following procedures:

Uninstall Procedure for WebSphere (Manual)

1. Make sure that you have the *WAS_HOME* and *JAVA_HOME* environment variables set to the WebSphere Home directory:
   - Solaris: `/opt/Websphere/AppServer`
   - AIX: `/usr/Websphere/AppServer`
   - Windows NT: `C:\Websphere\AppServer`

   and the Java home directory that resides in WebSphere

   - Solaris: `/opt/Websphere/AppServer/java`
   - AIX: `/usr/Websphere/AppServer/java`
   - Windows NT/2000: `C:\Websphere\AppServer\java`).
2. Open the Admin Client Java application, go to the **Topology** tab, expand **Application Node**, then expand **Default Server**.
3. Stop the default server by right-clicking and choosing **Stop**.
4. Under **Default Server** expand **Servlet Engine**.
5. Right-click on **IWWS**, then click on **Remove**. This will remove the `Repository` servlet.
6. Shut down the WebSphere Application server and ensure that all the Java processes associated with WebSphere are finished.
7. Test to make sure the `IWWS` servlet is not running by attempting to load the following URL: `http://websphere_server_name:port_number/iwws/repository`. You should see a "page not found" (404) error.
8. Navigate to the WebSphere home directory, remove the `iwws` directory, and remove the `iwws.properties` file that resides in the property directory.
9. Start up WebSphere Application Server, and the servlet engine should contain no `IWWS` servlet.

10. To uninstall the Enterprise Information Portal & Content Manager component, refer to the *Interwoven Turbo Product Guide for IBM Enterprise Information Portal and Content Manager* for details.

11. To uninstall the WebSphere Personalization component, refer to the *Interwoven Turbo Product Guide for IBM WebSphere Personalization* for details.

Uninstall Procedure for WebSphere (Auto)

1. The uninstall program is case-sensitive and information must be entered in exactly the form that the program asks for. For example, `y/n?` should be answered with **`y`** or **`n`**, not **`Y`** or **`N`**.

2. Make sure you have the *`WAS_HOME`* and *`JAVA_HOME`* environment variables set to the WebSphere Home directory
   - Solaris: `/usr/Websphere/AppServer`,
   - AIX: `/opt/Websphere/AppServer`
   - Windows NT: `C:/Websphere/AppServer`)
   - and the Java home directory that resides in WebSphere.

3. Browse to the location of the directory created during the installation package extraction.

4. Execute `uninstall.bat` (on Windows NT) or `uninstall.sh` (on AIX/Solaris).

5. To remove the WebSphere Application Server, WebSphere Personalization and Enterprise Information Portal & Content Manager components (or the Application Server or Enterprise Information Portal & Content Manager only), type **`y`** or just press **Enter** to confirm the uninstallation. To uninstall either the WebSphere Personalization or Enterprise Information Portal & Content Manager component only, enter **`n`**. Note that if the WebSphere Personalization or Enterprise Information Portal & Content Manager component is present, the user is not allowed to uninstall the Application Server while keeping WebSphere Personalization or Enterprise Information Portal & Content Manager. The default value is `y`.

6. If `n` was entered in step 5 to uninstall the WebSphere Personalization component, refer to the *Interwoven Turbo Product Guide for IBM WebSphere Personalization* for uninstall instructions. If `n` was entered in step 5 to uninstall the Enterprise Information Portal & Content Manager component, refer to the *Interwoven Turbo Product Guide for IBM Enterprise Information Portal and Content Manager* for uninstall instructions.
7. Enter the WebSphere Application Server root directory.

Uninstall Procedure for TeamSite (Manual)

1. Stop `iwproxy`.
2. Remove `sandboxpool.bat` and `sandboxpool.cgi` from `/iw-home/httpd/bin` (*`iw-home`* is the TeamSite home directory).
3. Copy `iw.cfg.iwws.bak` to `iw.cfg`. This will restore the original TeamSite settings.
4. If the Enterprise Information Portal & Content Manager component is installed, refer to the *Interwoven Turbo Product Guide for IBM Enterprise Information Portal and Content Manager* for uninstall instructions.
5. If the WebSphere Personalization component is installed, refer to the *Interwoven Turbo Product Guide for IBM WebSphere Personalization* for uninstall instructions.
6. Restart `iwproxy`.

Uninstall Procedure for TeamSite (Auto)

1. The uninstall program is case-sensitive and information must be entered in exactly the form that the program asks for. For example, `y/n?` should be answered with **`y`** or **`n`**, not **`Y`** or **`N`**.
2. Start the uninstall program for the WebSphere Application Server component by pressing **Enter** or typing **`y`**. The default value is `y`.
3. Make sure you have the *`TS_HOME`* and *`JAVA_HOME`* environment variables set to the TeamSite home directory (Solaris: `/usr/iw-home`, Windows NT: `C:/iw-home`) and your Java home directory (for example, Windows NT: `C:\jdk1.2.2\`).
4. Browse to the location of the directory created during the installation package extraction.

5. On Solaris, add execute permission to `install.sh` by running the command **`chmod +x install.sh`**.
6. Execute `uninstall.bat` (on Windows NT) or `uninstall.sh` (on AIX/Solaris).
7. Enter the TeamSite home directory (where TeamSite is installed).
8. Enter the location of the `iw.cfg` file.
9. If the Enterprise Information Portal & Content Manager component is installed, type **`y`** or press **Enter**. Refer to the *Interwoven Turbo Product Guide for IBM Enterprise Information Portal and Content Manager* for uninstall instructions.
10. If the WebSphere Personalization component is installed, type **`y`** or press **Enter**. Refer to the *Interwoven Turbo Product Guide for IBM WebSphere Personalization* for uninstall instructions.

Chapter 4

Using the Sandbox Pool Within TeamSite

Users can link their TeamSite workareas, staging areas or editions to a sandbox by accessing the Sandbox Pool menu item from the TeamSite GUI. If a sandbox is not linked to the current workarea then a check-out form is presented to the user. If a sandbox is linked to the current workarea then a check-in form is presented to the user.

Sandbox Check-Out Process

1. Log in to TeamSite and navigate to a workarea.

2. Select **File > Sandbox Pool** from within TeamSite. If a sandbox is already allocated to the current workarea, go to the section titled "Sandbox Check-In Process" below.

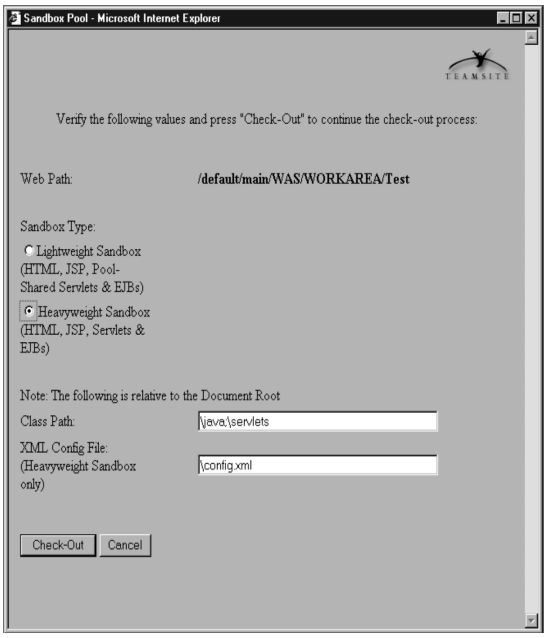
3. From the Sandbox Pool form, select a *lightweight* or *heavyweight* sandbox to check out.

4. In the `Classpath` field, enter the path, relative to the workarea where the Java classes will be located. If multiple classpaths are used, on aWindows NT platform separate them with `;`. For example, **`\java;\servlet`**. On Solaris or AIX platforms, separate them with `:`.

5. In the `XML Config File` field, enter the path and filename, relative to the workarea where the XML configuration file detailing the WebSphere Web application components is located.

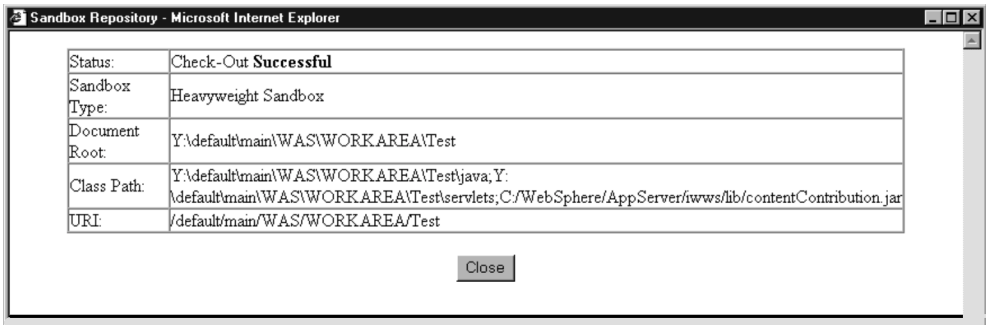
Note: This file is only necessary for the heavyweight sandbox since it can have a configuration different from any other sandbox. For more information on how to create the configuration XML, refer to the XML Configuration section.

6. Click the **Check-Out** button to check out the sandbox and link it to the current workarea.



Check Out Screen

7. A status page will be displayed with the status of the attempted check out. Click the **Close** button to remove the Sandbox Pool status page.



Check Out Confirmation

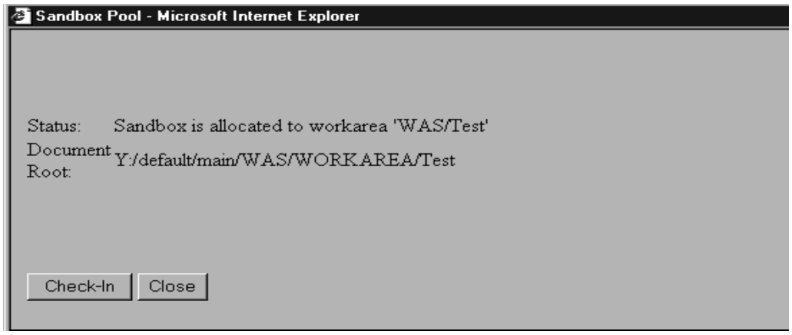
Note: Depending on the speed and amount of activity on the WebSphere server, it could take several minutes for the WebSphere Application Server environment to start the sandbox Web application. Attempting to access documents in the TeamSite environment immediately after check out could produce errors from the WebSphere Application Server; try back every few seconds. If the document is not served in a reasonable amount of time then check with an administrator for help.

Sandbox Check-In Process

1. Log in to TeamSite and navigate to a workarea.

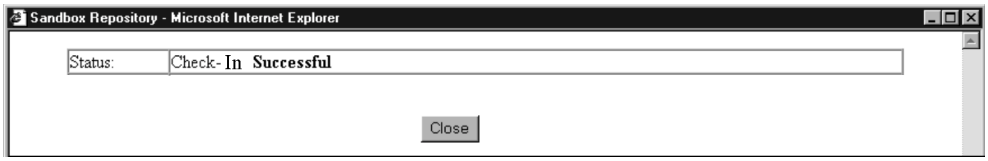
2. Select **File > Sandbox Pool** from within TeamSite. If a sandbox is not allocated to your current workarea then refer to the preceding section titled "Sandbox Check-Out Process."

3. You will be prompted with a form containing information on the sandbox that is linked to the current workarea. To release the sandbox from this workarea, click the **Check-In** button.



Check In Screen

4. A status page will be displayed with the status of the attempted check out. Click the **Close** button to remove the Sandbox Pool status page.



Check In Confirmation

Chapter 5

XML Configuration

For the WebSphere Application Server component of the Turbo product to run servlets and EJBs, the WebSphere environment must be updated to reflect the properties of the individual servlets and EJBs. An XML configuration file can be imported to update the WebSphere environment. This file may be custom-created, or a template may be exported from an existing application server inside of WebSphere Application Server.

The XML configuration file can be customized and used to perform different actions with the application server, Web application, EJB container, servlet engine, and so on. Actions may include `locate`, `update`, `create`, `delete`, `stop`, `start`, and `restart`. The various XML elements are associated with different actions. For details on how to perform these actions using the XML configuration file or how to export a Web application, consult the *IBM WebSphere Application Server* manual or Chapter 21 in the *IBM WebSphere Application Server V3.5 Handbook* on using the XML Config Utility.

XML Configuration File Components

The XML configuration file is a representation of the WebSphere topology tree. The file's hierarchy is similar to the administrative console's topology tree.

`<websphere-sa-config>` This element contains the whole XML document.
  `<node>` Represents the computer where the WebSphere Application Server instance is installed.
    `<application-server>` An application servers contained within the node.
      `<container>` Represents the EJB container within a particular app. server. (Within the container, EJB, DB datasource, user name, password, and other properties can be deployed.)
      `<ejb>` Represents an individual EJB and its properties.
      `</ejb>`
      `</container>`

      `<servlet-engine>` Represents a container where Web applications are created and kept.

```
        <web-application>  Represents one of the Web applications contained within  the servlet
                           engine, it contains document root, servlet classpath, web app alias path,
                           error pages, and so on.
              <servlet>  Represents an individual servlet and its properties.
              </servlet>
           </web-application>
        </servlet-engine>

    </application-server>
  </node>
</websphere-sa-config>
```

XML Configuration Examples

Enabling a servlet to run by registering it in the XML configuration file

The following example XML configuration file will locate the node and application server and create a Web application containing three servlets (the first two sevlets were exported from a default Web application created within the Administrator console). The Web application should have at least the `File Serving` and `JSP` servlets to run JSP pages and other files. It is also recommended that an error handler should be created, whether it is a JSP error page or the default `ErrorReporter` servlet that comes with WebSphere Application Server. In this case, a JSP page named `debug.jsp` is created, stored in the document root, and is associated with the `<error-page>` tag. The `<auto-reload>` and `<reload-interval>` tags are used to enable the automatic detection and refreshing of servlet changes within a specified period of time (in seconds). Therefore, if a servlet is changed (re-compiled), it is not necessary to stop and restart the Web application to flush the servlet cache. The `HelloServlet` servlet will be registered and created during heavyweight sandbox allocation or lightweight sandbox pool creation when used with the following XML configuration file. The values enclosed in square brackets `[]` will be substituted at runtime by the sandbox pool. In this scenario, other servlets cannot be run within this Web application, as they have not been registered.

Note: If a property does not exist and an update action is associated with it, the update action will create the property. On a Windows NT system, please ensure that neither `"\"` nor `"\\"` is used for any file path separator because the XML Config Utility can handle `"/"` for all platforms.

```
<websphere-sa-config>
  <node name="[node name]" action="locate">
    <application-server name="[application server name]" action="[update]">

      <servlet-engine name="[servlet engine name]" action="[locate]">
        <web-application name="[web application name]" action="[create]">
          <description>example servlets</description>
          <document-root>[document root]</document-root>
          <classpath>
            <path value="[servlet classpath]"/>
          </classpath>
          <error-page> C:/WebSphere/AppServer/hosts/default_host/examples/web/
debug_error.jsp</error-page>
          <filter-list/>
          <group-attributes/>
          <auto-reload>true</auto-reload>
          <reload-interval>3000</reload-interval>
          <enabled>true</enabled>
          <root-uri>[default_host/webapp/examples]</root-uri>
          <shared-context>false</shared-context>
          <shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>

          <servlet name="File Serving Enabler" action="create">
            <description>Auto-Generated - File Serving Servlet</description>
            <code>com.ibm.servlet.engine.webapp.SimpleFileServlet</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="JSP 1.0 Processor" action="create">
            <description>Auto-Generated - Generates JSP 1.0 output</description>
            <code>com.sun.jsp.runtime.JspServlet</code>
            <init-parameters>
              <parameter name="workingDir" value="C:/WebSphere/AppServer/hosts/
default_host/WSsamples_app/web"/>
            </init-parameters>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="*.jsp"/>
```

```
            <uri value="*.jsv"/>
            <uri value="*.jsw"/>
          </uri-paths>
          <enabled>true</enabled>
        </servlet>

        <servlet name="HelloServlet" action="create">
          <description>Simple Hello Servlet</description>
          <code>HelloServlet</code>
          <init-parameters/>
          <load-at-startup>false</load-at-startup>
          <debug-mode>false</debug-mode>
          <uri-paths>
            <uri value="/servlet/HelloServlet"/>
          </uri-paths>
          <enabled>true</enabled>
        </servlet>

      </web-application>

      <session-manager name="Session Manager" action="create">
        <enable-sessions>true</enable-sessions>
        <enable-url-rewriting>false</enable-url-rewriting>
        <enable-cookies>true</enable-cookies>
        <enable-protocol-switch-rewriting>false</enable-protocol-switch-
rewriting>
        <cookie name="sesessionid">
          <comment>servlet session support</comment>
          <domain></domain>
          <maximum>-1</maximum>
          <path>/</path>
          <secure>false</secure>
        </cookie>
        <interval-invalidation-time>1800</interval-invalidation-time>
        <persistent-sessions>false</persistent-sessions>
        <persistence-type>directodb</persistence-type>
        <database location="jdbc:db2:was">
          <driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>
          <user-id></user-id>
          <password></password>
          <number-of-connections>30</number-of-connections>
        </database>
        <enable-stat-collection>true</enable-stat-collection>
        <using-cache>false</using-cache>
        <using-multi-row>false</using-multi-row>
```

```
          <using-manual-update>false</using-manual-update>
          <using-native-access>false</using-native-access>
          <base-memory-size>1000</base-memory-size>
          <allow-overflow>true</allow-overflow>
          <data-source></data-source>
        </session-manager>

        <user-profile-manager name="User Profile Manager" action="create">
          <enable-user-profile>false</enable-user-profile>
          <data-wrapper>com.ibm.servlet.personalization.userprofile.UserProfile</
data-wrapper>
          <remote-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnly</remote-interface-ro>
          <remote-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWrite</remote-interface-rw>
          <home-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome</home-interface-
ro>
          <home-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome</home-interface-
rw>
          <jndi-name-ro>UP_ReadOnlyHome</jndi-name-ro>
          <jndi-name-rw>UP_ReadWriteHome</jndi-name-rw>
        </user-profile-manager>

      </servlet-engine>
    </application-server>
  </node>
</websphere-sa-config>
```

Registering the Auto-Invoker servlet in the XML configuration file to call a servlet by class name

The following example XML configuration file will locate the node and application server and create a Web application containing four servlets. All of the servlets are exported from a default Web application created within the Administrator console. In this example, the IBM `ErrorReporter` servlet that comes with WebSphere Application Server is used to handle Web application errors and is associated with the `<error-page>` tag. The `<auto-reload>` and `<reload-interval>` tags are used to enable the automatic detection and refreshing of servlet changes within a specified period of time (in seconds). Therefore, if a servlet is changed (re-compiled), it is not necessary to stop and restart the Web application to flush the servlet cache. The `Auto-Invoker` servlet is created and

registered with the Web application, so that other servlets can be called by their class name within JSP pages or other servlets. There is no need to register or to create individual servlets and associate them with the Web application. The values enclosed in square brackets [ ] represent values that will be substituted at runtime by the sandbox pool. In this scenario, other servlets can be run within this Web application as they will be found by the `Auto-Invoker` servlet.

Note: If a property does not exist and an update action is associated with it, the update action will create the property. On a Windows NT system, please ensure that neither "\" nor "\\" is used for any file path separator because the XML Config Utility can handle "/" for all platforms.

```
<websphere-sa-config>
  <node name="[node name]" action="locate">
    <application-server name="[application server name]" action="[update]">

      <servlet-engine name="[servlet engine name]" action="[locate]">
        <web-application name="[web application name]" action="[create]">
          <description>example servlets</description>
          <document-root>[document root]</document-root>
          <classpath>
            <path value="[servlet classpath]"/>
          </classpath>
          <error-page>/ErrorReporter</error-page>
          <filter-list/>
          <group-attributes/>
          <auto-reload>true</auto-reload>
          <reload-interval>3000</reload-interval>
          <enabled>true</enabled>
          <root-uri>[default_host/webapp/examples]</root-uri>
          <shared-context>false</shared-context>
          <shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>

          <servlet name="Error Reporting Facility" action="create">
            <description>Auto-Generated - Default error reporter servlet</
description>
            <code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/ErrorReporter"/>
            </uri-paths>
            <enabled>true</enabled>
```

```
            </servlet>
            <servlet name="File Serving Enabler" action="create">
              <description>Auto-Generated - File Serving Servlet</description>
              <code>com.ibm.servlet.engine.webapp.SimpleFileServlet</code>
              <init-parameters/>
              <load-at-startup>true</load-at-startup>
              <debug-mode>false</debug-mode>
              <uri-paths>
                <uri value="/"/>
              </uri-paths>
              <enabled>true</enabled>
            </servlet>
            <servlet name="Auto-Invoker" action="create">
              <description>Auto-Generated - Serves Servlets by Classname</
  description>
              <code>com.ibm.servlet.engine.webapp.InvokerServlet</code>
              <init-parameters/>
              <load-at-startup>true</load-at-startup>
              <debug-mode>false</debug-mode>
              <uri-paths>
                <uri value="/servlet"/>
              </uri-paths>
              <enabled>true</enabled>
            </servlet>
            <servlet name="JSP 1.0 Processor" action="create">
              <description>Auto-Generated - Generates JSP 1.0 output</description>
              <code>com.sun.jsp.runtime.JspServlet</code>
              <init-parameters>
                <parameter name="workingDir"
  value="C:\WebSphere\AppServer\hosts\default_host\WSsamples_app\web"/>
              </init-parameters>
              <load-at-startup>true</load-at-startup>
              <debug-mode>false</debug-mode>
              <uri-paths>
                <uri value="*.jsp"/>
                <uri value="*.jsv"/>
                <uri value="*.jsw"/>
              </uri-paths>
              <enabled>true</enabled>
            </servlet>
          </web-application>

          <session-manager name="Session Manager" action="create">
            <enable-sessions>true</enable-sessions>
            <enable-url-rewriting>false</enable-url-rewriting>
```

```
          <enable-cookies>true</enable-cookies>
          <enable-protocol-switch-rewriting>false</enable-protocol-switch-
rewriting>
          <cookie name="sesessionid">
            <comment>servlet session support</comment>
            <domain></domain>
            <maximum>-1</maximum>
            <path>/</path>
            <secure>false</secure>
          </cookie>
          <interval-invalidation-time>1800</interval-invalidation-time>
          <persistent-sessions>false</persistent-sessions>
          <persistence-type>directodb</persistence-type>
          <database location="jdbc:db2:was">
            <driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>
            <user-id></user-id>
            <password></password>
            <number-of-connections>30</number-of-connections>
          </database>
          <enable-stat-collection>true</enable-stat-collection>
          <using-cache>false</using-cache>
          <using-multi-row>false</using-multi-row>
          <using-manual-update>false</using-manual-update>
          <using-native-access>false</using-native-access>
          <base-memory-size>1000</base-memory-size>
          <allow-overflow>true</allow-overflow>
          <data-source></data-source>
        </session-manager>

        <user-profile-manager name="User Profile Manager" action="create">
          <enable-user-profile>false</enable-user-profile>
          <data-wrapper>com.ibm.servlet.personalization.userprofile.UserProfile</
data-wrapper>
          <remote-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnly</remote-interface-ro>
          <remote-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWrite</remote-interface-rw>
          <home-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome</home-interface-
ro>
          <home-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome</home-interface-
rw>
          <jndi-name-ro>UP_ReadOnlyHome</jndi-name-ro>
          <jndi-name-rw>UP_ReadWriteHome</jndi-name-rw>
```

```
        </user-profile-manager>

      </servlet-engine>
    </application-server>
  </node>
</websphere-sa-config>
```

Performing EJB creation and deployment in the XML configuration file

The following example XML configuration file will locate the node, update the application server, and create a Web application containing four servlets and two EJBs. The servlets are exported from a default Web application created within the Administrator console. In this example, the IBM `ErrorReporter` servlet that comes with WebSphere Application Server is used to handle Web application errors and is associated with the `<error-page>` tag. The `<auto-reload>` and `<reload-interval>` tags are used to enable the automatic detection and refreshing of servlet changes within a specified period of time (in seconds). Therefore, if a servlet is changed (re-compiled), it is not necessary to stop and restart the Web application to flush the servlet cache. The `Auto-Invoker` servlet is created and registered with the Web application, so that other servlets can be called by their classname within JSP pages or other servlets. Also the `Increment` and `Hello` sample EJBs that come with WebSphere Application Server will be created and deployed. The `Hello` EJB will display a very simple message, and the `Increment` EJB will retrieve and update the count stored in the database. In this case, the XML configuration file needs to create an EJB container, the specified EJBs, and a datasource. This is the reason why the DB user name and password need to be supplied in the XML configuartion file. The values enclosed in square brackets `[]` represent values that will be substituted at runtime by the sandbox pool. In this scenario, other servlets can be run within this Web application, because they will be found by the `Auto-Invoker` servlet.

Note: If a property does not exist and an update action is associated with it, the update action will create the property. On a Windows NT system, please ensure that neither "\" nor "\\" is used for any file path separator since the XML Config Utility can handle "/" for all platforms.

```
<websphere-sa-config>
  <node name="[node name]" action="locate">
    <application-server name="[application server name]" action="[update]">

<container name="Default Container" action="update">
     <user-id>wsdemo</user-id>
     <password>wsdemo1</password>
```

```
    <data-source name="sample" />
    <ejb name="HelloEJB" action="create">
       <jar-file>C:/WebSphere/AppServer/deployableEJBs/Hello.jar</jar-file>
       <home-name>HelloHome</home-name>
       <user-id></user-id>
       <password></password>
       <create-db-table>false</create-db-table>
       <find-for-update>false</find-for-update>
       <minimum-pool-size>5</minimum-pool-size>
       <maximum-pool-size>500</maximum-pool-size>
       <primary-key-check>false</primary-key-check>
       <db-exclusive-access>false</db-exclusive-access>
    </ejb>
    <ejb name="Increment" action="create">
       <jar-file>C:/WebSphere/AppServer/deployableEJBs/Increment.jar</jar-file>
       <home-name>Increment</home-name>
       <user-id></user-id>
       <password></password>
       <create-db-table>true</create-db-table>
       <find-for-update>true</find-for-update>
       <minimum-pool-size>5</minimum-pool-size>
       <maximum-pool-size>500</maximum-pool-size>
       <primary-key-check>true</primary-key-check>
       <db-exclusive-access>false</db-exclusive-access>
    </ejb>
</container>

     <servlet-engine name="[servlet engine name]" action="[locate]">
       <web-application name="[web application name]" action="[create]">
         <description>example servlets</description>
         <document-root>[document root]</document-root>
         <classpath>
           <path value="[servlet classpath]"/>
         </classpath>

         <error-page>/ErrorReporter</error-page>
         <filter-list/>
         <group-attributes/>
         <auto-reload>true</auto-reload>
         <reload-interval> 3000 </reload-interval>
         <enabled>true</enabled>
         <root-uri>[default_host/webapp/examples]</root-uri>
         <shared-context>false</shared-context>
         <shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>
```

```
          <servlet name="Error Reporting Facility" action="create">
            <description>Auto-Generated - Default error reporter servlet</
description>
            <code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/ErrorReporter"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="File Serving Enabler" action="create">
            <description>Auto-Generated - File Serving Servlet</description>
            <code>com.ibm.servlet.engine.webapp.SimpleFileServlet</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="Auto-Invoker" action="create">
            <description>Auto-Generated - Serves Servlets by Classname</
description>
            <code>com.ibm.servlet.engine.webapp.InvokerServlet</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/servlet"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="JSP 1.0 Processor" action="create">
            <description>Auto-Generated - Generates JSP 1.0 output</description>
            <code>com.sun.jsp.runtime.JspServlet</code>
            <init-parameters>
              <parameter name="workingDir" value="C:/WebSphere/AppServer/hosts/
default_host/WSsamples_app/web"/>
            </init-parameters>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
```

```
              <uri value="*.jsp"/>
              <uri value="*.jsv"/>
              <uri value="*.jsw"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>


        </web-application>
        <session-manager name="Session Manager" action="update">
          <enable-sessions>true</enable-sessions>
          <enable-url-rewriting>false</enable-url-rewriting>
          <enable-cookies>true</enable-cookies>
          <enable-protocol-switch-rewriting>false</enable-protocol-switch-
rewriting>
          <cookie name="sesessionid">
            <comment>servlet session support</comment>
            <domain></domain>
            <maximum>-1</maximum>
            <path>/</path>
            <secure>false</secure>
          </cookie>
          <interval-invalidation-time>1800</interval-invalidation-time>
          <persistent-sessions>false</persistent-sessions>
          <persistence-type>directodb</persistence-type>
          <database location="jdbc:db2:was">
            <driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>
            <user-id></user-id>
            <password></password>
            <number-of-connections>30</number-of-connections>
          </database>
          <enable-stat-collection>true</enable-stat-collection>
          <using-cache>false</using-cache>
          <using-multi-row>false</using-multi-row>
          <using-manual-update>false</using-manual-update>
          <using-native-access>false</using-native-access>
          <base-memory-size>1000</base-memory-size>
          <allow-overflow>true</allow-overflow>
          <data-source></data-source>
        </session-manager>
        <user-profile-manager name="User Profile Manager" action="update">
          <enable-user-profile>false</enable-user-profile>
          <data-wrapper>com.ibm.servlet.personalization.userprofile.UserProfile</
data-wrapper>
```

```
          <remote-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnly</remote-interface-ro>
          <remote-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWrite</remote-interface-rw>
          <home-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome</home-interface-
ro>
          <home-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome</home-interface-
rw>
          <jndi-name-ro>UP_ReadOnlyHome</jndi-name-ro>
          <jndi-name-rw>UP_ReadWriteHome</jndi-name-rw>
        </user-profile-manager>
      </servlet-engine>
    </application-server>
  </node>
</websphere-sa-config>
```

Setting up to run shared and deployed EJBs in XML config file

The following example XML configuration file will locate the node, update the application server, and create a Web application containing four servlets. The servlets are exported from a default Web application created within the Administrator console. In this example, the IBM `ErrorReporter` servlet that comes with WebSphere Application Server is used to handle Web application errors and is associated with the `<error-page>` tag. The `<auto-reload>` and `<reload-interval>` tags are used to enable the automatic detection and refreshing of servlet changes within a specified period of time (in seconds). Therefore, if a servlet is changed (re-compiled), it is not necessary to stop and restart the Web application to flush the servlet cache. The `Auto-Invoker` servlet is created and registered with the Web application, so that other servlets can be called by their classname within JSP pages or other servlets. In order for other JSP or EJB developers to run and access the deployed `Hello` and `Increment` EJBs on the sandboxes, they need to provide the classpath for deployed EJB JAR files as the command-line argument in the configuration file. This assumes that the EJBs already exist in another application server container and that heavyweight and lightweight sandboxes are created and located under the same node (same server machine) and they all deployed to the same directory (the default directory is `C:\`*`WAS_HOME`*`\deployedEJBs`).

In this particular example, heavyweight sandbox users will be able to use any deployed EJB by performing a sandbox allocation action with the XML configuration file below. The values enclosed

in square brackets [ ] represent values that will be substituted at runtime by the sandbox pool. In this scenario, other servlets can be run within this Web application as they will be found by the `Auto-Invoker` servlet.

Note: If a property does not exist and an update action is associated with it, the update action will create the property. On a Windows NT system, please ensure that neither "\" nor "\\" is used for any file path separator because the XML Config Utility can handle "/" for all platforms.

For lightweight sandbox users, there are two ways to achieve this assuming that heavyweight and lightweight sandboxes are created and located under the same node (same server computer) and they all deployed to the same directory (The default directory is `C:\`*`WAS_Home`*`\deployedEJBs`). Before performing this operation, make sure all lightweight sandbox users are offline and there is no sandbox activity, because the lightweight sandbox pool will be unavailable.

- Destroy and create the lightweight sandbox pool with the following XML configuration file.
- Without destroying and recreating the lightweight sandbox pool, the WebSphere system administrator can use the Administrator console to add the full path of those deployed EJB JAR files to the command-line argument field. To perform this, open the Admin Client, go to the **Topology** tab, expand **Application Node**, then click on **Sandbox Pool**. The property sheet should be displayed on the right panel. Now go to the **Command-Line Arguments** text box, and type the following as one line:
  `-classpath C:/WebSphere/AppServer/deployedEJBs/Hello.jar;C:/WebSphere/AppServer/deployedEJBs/Increment.jar`

After that, stop the Sandbox Pool application server, and re-start it. The XML Config Utitlity that comes with WebSphere Application Server can also perform this operation.

```
 <websphere-sa-config>
  <node name="[node name]" action="locate">
    <application-server name="[application server name]" action="[update]">

      <command-line-arguments>
<arg>-classpath</arg>
<arg>C:/WebSphere/AppServer/deployedEJBs/Hello.jar;C:/WebSphere/AppServer/
deployedEJBs/Increment.jar</arg>
      </command-line-arguments>
```

```
      <servlet-engine name="[servlet engine name]" action="[locate]">
        <web-application name="[web application name]" action="[create]">
          <description>example servlets</description>
          <document-root>[document root]</document-root>
          <classpath>
            <path value="[servlet classpath]"/>
          </classpath>

          <error-page>/ErrorReporter</error-page>
          <filter-list/>
          <group-attributes/>
          <auto-reload>true</auto-reload>
          <reload-interval> 3000 </reload-interval>
          <enabled>true</enabled>
          <root-uri>[default_host/webapp/examples]</root-uri>
          <shared-context>false</shared-context>
          <shared-context-jndi-name>SrdSrvltCtxHome</shared-context-jndi-name>

          <servlet name="Error Reporting Facility" action="create">
            <description>Auto-Generated - Default error reporter servlet</
description>
            <code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/ErrorReporter"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="File Serving Enabler" action="create">
            <description>Auto-Generated - File Serving Servlet</description>
            <code>com.ibm.servlet.engine.webapp.SimpleFileServlet</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="Auto-Invoker" action="create">
            <description>Auto-Generated - Serves Servlets by Classname</
description>
```

```
            <code>com.ibm.servlet.engine.webapp.InvokerServlet</code>
            <init-parameters/>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="/servlet"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>
          <servlet name="JSP 1.0 Processor" action="create">
            <description>Auto-Generated - Generates JSP 1.0 output</description>
            <code>com.sun.jsp.runtime.JspServlet</code>
            <init-parameters>
              <parameter name="workingDir" value="C:/WebSphere/AppServer/hosts/
default_host/WSsamples_app/web"/>
            </init-parameters>
            <load-at-startup>true</load-at-startup>
            <debug-mode>false</debug-mode>
            <uri-paths>
              <uri value="*.jsp"/>
              <uri value="*.jsv"/>
              <uri value="*.jsw"/>
            </uri-paths>
            <enabled>true</enabled>
          </servlet>


        </web-application>
        <session-manager name="Session Manager" action="update">
          <enable-sessions>true</enable-sessions>
          <enable-url-rewriting>false</enable-url-rewriting>
          <enable-cookies>true</enable-cookies>
          <enable-protocol-switch-rewriting>false</enable-protocol-switch-
rewriting>
          <cookie name="sesessionid">
            <comment>servlet session support</comment>
            <domain></domain>
            <maximum>-1</maximum>
            <path>/</path>
            <secure>false</secure>
          </cookie>
          <interval-invalidation-time>1800</interval-invalidation-time>
          <persistent-sessions>false</persistent-sessions>
          <persistence-type>directodb</persistence-type>
          <database location="jdbc:db2:was">
```

```
            <driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>
            <user-id></user-id>
            <password></password>
            <number-of-connections>30</number-of-connections>
          </database>
          <enable-stat-collection>true</enable-stat-collection>
          <using-cache>false</using-cache>
          <using-multi-row>false</using-multi-row>
          <using-manual-update>false</using-manual-update>
          <using-native-access>false</using-native-access>
          <base-memory-size>1000</base-memory-size>
          <allow-overflow>true</allow-overflow>
          <data-source></data-source>
        </session-manager>
        <user-profile-manager name="User Profile Manager" action="update">
          <enable-user-profile>false</enable-user-profile>
          <data-wrapper>com.ibm.servlet.personalization.userprofile.UserProfile</
data-wrapper>
          <remote-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnly</remote-interface-ro>
          <remote-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWrite</remote-interface-rw>
          <home-interface-
ro>com.ibm.servlet.personalization.userprofile.UP_ReadOnlyHome</home-interface-
ro>
          <home-interface-
rw>com.ibm.servlet.personalization.userprofile.UP_ReadWriteHome</home-interface-
rw>
          <jndi-name-ro>UP_ReadOnlyHome</jndi-name-ro>
          <jndi-name-rw>UP_ReadWriteHome</jndi-name-rw>
        </user-profile-manager>
      </servlet-engine>
    </application-server>
  </node>
</websphere-sa-config>
```

Chapter 6

Troubleshooting

Included in this section are potential problems you may encounter when installing and using the TeamSite and WebSphere components of the Turbo product. Potential problems include the following:

- Attempting to create a light sandbox pool on Solaris, but the command-line arguments are too long for the shell.

  Before you reach the character limit for one line, enter a **** and a carriage return (with no space at the end of each line) to inform the shell you will be continuing on another line.

- After checking out a sandbox, you cannot view content through the TeamSite GUI.
  - Proxy settings may not be configured correctly. The proxy must be configured to remap requests from the TeamSite server to WebSphere. See the "Proxy Configuration" section in Chapter 3 for suggestions how to do this.

    Tip: Verify that the Web application in WebSphere Application Server is configured correctly (Web path, classpath, doc root, and so on). Try to access the page directly from the browser, without going through the TeamSuite GUI and proxy.

  - WebSphere may not process the request if the host name is not in the Host Aliases list. For the request `http://example.com/default/main/Website/WORKAREA/test/index.html`, `example.com` is the host name. Verify that `example.com` appears in the Host Aliases list of the **Virtual Host > Advanced** tab in the WebSphere Application Server administration client.
  - The `file` or `jsp` servlets may be missing from the `config.xml` file. See Chapter 5, "XML Configuration" for details about creating an example XML configuration file.
  - Load on the WebSphere Application Server can slow the processing of initial sandbox allocation requests. If the content is not immediately available, try accessing the content a few minutes after the sandbox has been checked out.

- No sandboxes are available when trying to check out a sandbox.

  Check to ensure that the sandbox pool has been created and that a sandbox is available. A heavyweight sandbox will be a WebSphere Application Server starting with "Sandbox" and ending with an integer. If the Web application underneath the application server carries the same name as the application server, then an available sandbox exists. A lightweight sandbox will be a Web application starting with "Sandbox" and ending with an integer. All lightweight sandboxes will be located underneath a WebSphere Application Server named "Sandbox Pool." If the Web application (sandbox) has a name starting with Sandbox, then it is available for check out.

- Cannot locate `config.xml`.

  Permission settings may not allow access to the file. Check to see that permissions allow reading of the `config.xml` file.

Appendix A

Limitations and Assumptions

General

- This document does not provide any information on how to use the WebSphere Application Server. There is no discussion on the best practices for using WebSphere Application Server nor directions on its usage of servlets, JSPs, EJBs, and Web applications. Refer to the *IBM WebSphere Application Server 3.5 Handbook* for details on these topics.
- The Turbo product is designed to support a configuration of one instance of TeamSite with one instance of WebSphere Application Server (a single node). This is because the original design is based on the capability of WebSphere Application Server to support the multiple application servers and multiple Web applications within one application server on a single instance of WebSphere Application Server.
- This Turbo product will only work if security is disabled on the administrative console client.
- Any file path separator specified in the `config.xml` file used by this integration needs to be `/` instead of `\` on the Windows NT platform because WebSphere Application Server will perform the proper conversion based on the appropriate platform. Problems may occur if `\` is used instead of `/` in the `config.xml` file on the Windows NT platform.

Servlets

Both lightweight and heavyweight sandboxes can be used by JSP developers to dynamically test changes for servlet development without restarting the Web application corresponding to their workareas. This will work "out of the box" as long as the `<auto-reload>` and `<reload-interval>` tags are supplied in the `config.xml` file during the sandbox allocation process. For more information, please refer to the *IBM WebSphere Application Server 3.5 Handbook*.

EJBs

When an EJB is created and deployed in an application server container within WebSphere Application Server, a special JAR file named `_DeployedEJBName.jar` will be created in the *`WAS_ROOT`*`/deployedEJBs` directory. This directory serves as a centralized repository for all deployed EJB packages. The Java Naming and Directory Interface (JNDI) naming service for WebSphere Application Server is global to every application server and unique name constraint is applied to each EJB. This rule also applies to other components accessed through a JNDI naming service, like DataSource.

Multiple EJB developers will not be able to deploy the same EJB from their workareas at the same time, because their deployment will be landing on the same JAR directory and the first deployed JAR file will always get overwritten by the latter deployment. This rule applies to use and share testing EJB by QA team as well.

The following are some scenarios that might help facilitate using this for EJB development and the QA cycle. All of the scenarios assume that one version of `HelloEJB` was already deployed and other users are sharing this `HelloEJB` through the deployed JAR file, which is part of the application server classpath parameter automatically set up in their workarea configuration file during heavyweight sandbox allocation or lightweight sandbox creation. These scenarios demonstrate how only one `HelloEJB` can be deployed and used at a time

- **Scenario 1**: The QA team is testing a build which contains the latest version of `HelloEJB` in the QA Workarea, while developer A in Workarea A is fixing bugs on top of the latest version of `HelloEJB`. In this scenario, the QA team has specified all necessary EJBs and servlets for creation and deployment in the configuration file used by heavyweight sandbox allocation. Developer A should not deploy the newly compiled EJB because this will overwrite the build version of `HelloEJB` that is already deployed.

- **Scenario 2**: Developers working in different workareas. In this scenario, developer A in Workarea A created and deployed the latest version of `HelloEJB` during heavyweight sandbox allocation. Developer B in Workarea B is using this latest `HelloEJB` because it was deployed in the app server classpath during sandbox allocation. Developer A made some changes to the `HelloEJB` and wants to redeploy it. If Developer A deployed the HelloEJB, developer B would be impacted because developer A's new deployed JAR file would overwrite the existing deployed EJB JAR file.

- **Scenario 3**: Developer A created and deployed the latest version of `HelloEJB` during heavyweight sandbox allocation. Developer B used this latest `HelloEJB` which resides in developer A's application server container, by providing the deployed JAR file location in the application server classpath. Developer A finished testing and deallocated the sandbox from his workarea. Upon sandbox deallocation, the `HelloEJB` residing in the EJB container for this sandbox will be destroyed. Developer B will no longer be able to use `HelloEJB` and will be getting an exception from WebSphere.

In summary, build and QA environments should have their own application servers for testing. This would permit testing to occur on a version of the code that is closer to production than what is currently being developed by the engineers. This will also prevent development activities from interfering with the QA process. Business processes exist for using TeamSite workflow and deployment products to manage the migration of code from developement and test environments to the QA environment.