



**I N T E R W O V E N**

# **Interwoven® Turbo Product Guide**

**v2.0**

**for IBM® WebSphere® Personalization**

Copyright 2001 Interwoven, Inc. All rights reserved.

No part of this publication (hardcopy or electronic form) may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Interwoven. Information in this manual is furnished under license by Interwoven, Inc. and may only be used in accordance with the terms of the license agreement. If this software or documentation directs you to copy materials, you must first have permission from the copyright owner of the materials to avoid violating the law, which could result in damages or other remedies.

Interwoven, TeamSite, OpenDeploy and the logo are registered trademarks of Interwoven, which may be registered in certain jurisdictions. SmartContext, Content Express, TeamXpress, DataDeploy, the tagline and service mark are trademarks of Interwoven, Inc. which may be registered in certain jurisdictions. All other trademarks are owned by their respective owners.



**I N T E R W O V E N**

Interwoven, Inc.

1195 West Fremont Ave.

Sunnyvale, CA 94087

<http://www.interwoven.com>

Printed in the United States of America

Version 2.0

Part #40-10-40-42-04-200-601

# Table of Contents

<b>Chapter 1: About This Product</b>	<b>5</b>
What is an Interwoven Turbo Product?	5
Features of the Turbo Product Family	6
Application Server Component	6
Personalization Server Component	6
Business-User Support	7
Architectural Overview	7
General Turbo Architecture	7
Application Servers and Virtualization	9
Who Can Use This Product?	10
Business Users	11
Templating	11
Metadata Capture	11
Workflow Approval	11
Application Developers	12
Presentation Layer Developers	12
About This Document	13
<b>Chapter 2: Introduction</b>	<b>15</b>
Overview	15
Prerequisite Software	15
Development Environment	15
Database Server Host	15
WebSphere Server Host	15
TeamSite Server Host	16
Production Environment	16
Database Server Host	16
WebSphere Server Host	17
Client	17
Prerequisite Hardware	17
TeamSite Host Configuration	17
WebSphere Application Server Host Configuration	17
Product Architecture	18
Data Capture	20
Data Preview (phase 1)	20



- Data Transfer 20
- Data Preview (phase 2) 21
- Migration to Production 22

## **Chapter 3: Installation and Configuration 23**

- Overview 23
- Installing Turbo for WebSphere Personalization on the WebSphere Application Server Host 23
  - Terms 23
  - Before You Begin 24
  - Installation 24
  - Post-Installation Checks 26
- Updating the SAMPLE Database 27
- Installing Turbo for WebSphere Personalization on the TeamSite Server 32
  - Before You Begin 32
  - Installation 32
  - Post-Installation Checks 35
- Installing and Configuring the iwatsub Command Trigger 38
- Uninstalling Turbo for WebSphere Personalization on the WebSphere Application Server Host 39
  - Before You Begin 39
  - Uninstallation 39
- Uninstalling Turbo for WebSphere Personalization on the TeamSite Server 41

## **Chapter 4: Using the Sample Application 43**

- Overview 43
- Publishing the Sample Application into the Workarea 43
  - Important Notes on Publishing the Sample Application 48
    - Webapp Web Path 48
    - Style of Links 49
- Creating Content for the Sample Application through TeamSite Templating 50
  - Points to Remember 55
    - The Title Value in the Companynews and Articles Tables 55
    - A Limitation of the News Page Rules 57

## **Chapter 5: Customization 59**

- Overview 59
- Customizing Data Categories and Data Types 59
- Customizing datacapture.cfg Files 61
- Customizing Presentation Templates 64
- Customizing Metadata Rules 66

Customizing Metadata Capture Templates	67
Customizing Resource Mapping Files	68
Customizing the iwwpe.cfg Configuration File	69
Diagram Key	71
Customizing the email_map.cfg Configuration File	71

## **Chapter 6: Deploying the Sample Application 75**

Overview	75
Before You Begin	75
Server Components Setup	76
Client Components Setup	77
Server Components Configuration	78
iwwpe_remote_receive.cfg	78
iwwpe_xml_db.cfg	78
The pznload Utility	79
pznResourcesToLoad.txt	79
pznRulesToLoad.txt	79
pznload Utility Usage	80
iwwpe_backup_data.sql	81
iwwpe_restore_data.sql	81
Client Components Configuration	82
iwwpe_local_send.cfg	82
deployment=deploy_jsp	82
deployment=deploy_classes	83
deployment=deploy_db_xml	83
deployment=deploy_rules	85
deployment=deploy_resources	86
iwwpe_dump_load.pl Usage	86
iwwpe_db_xml.cfg	87
OpenDeploy Invocation	87
How It Works	88



INTERWOVEN

## 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.



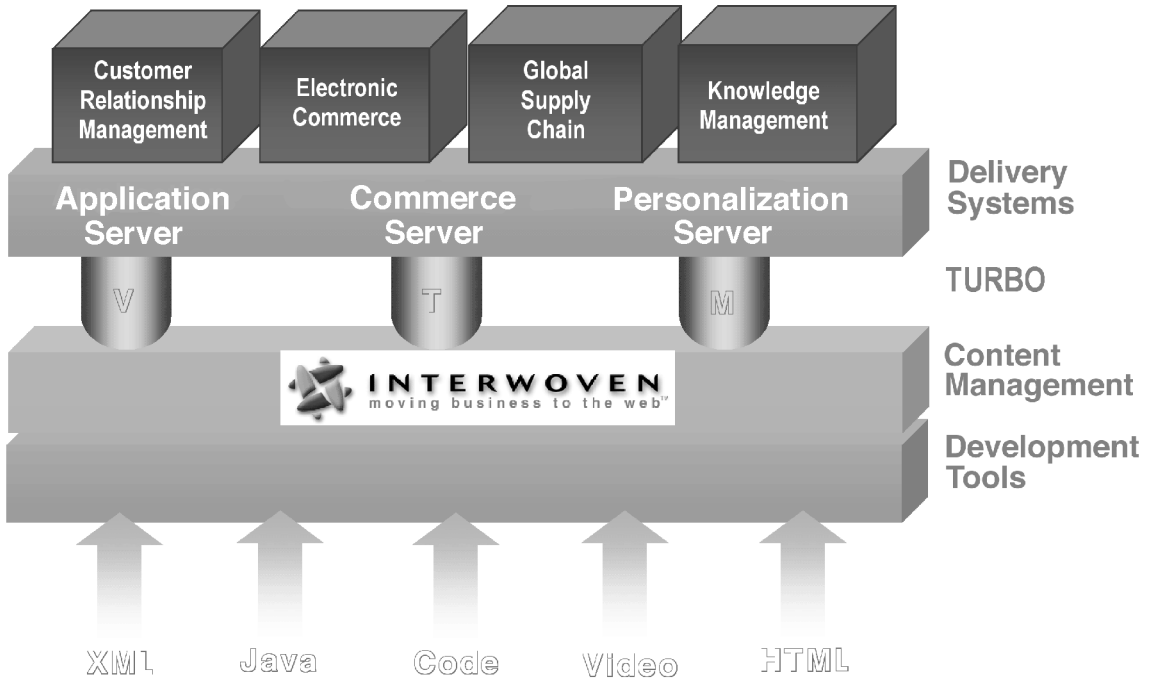
## **Business-User Support**

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**

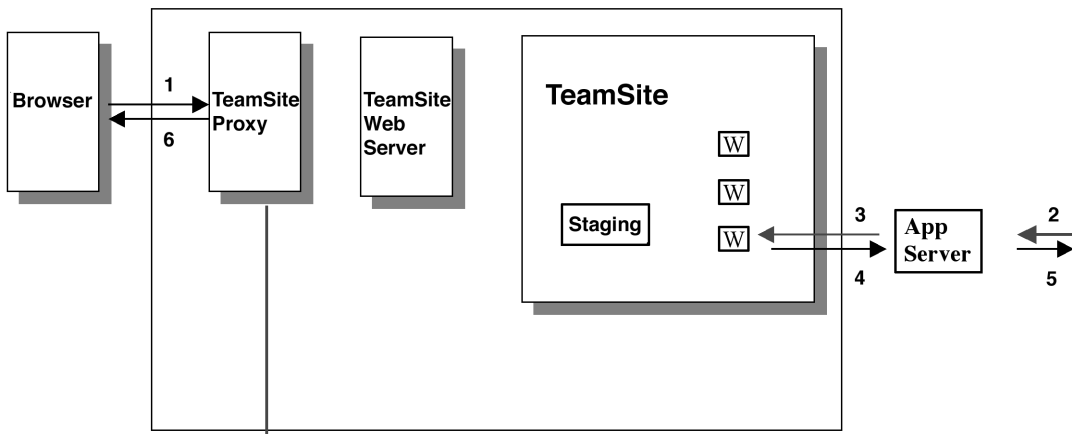
Turbo connects the TeamSite content management functionality to various delivery systems while augmenting the TeamSite native virtualization, templating and metadata capabilities. Development tools and delivery systems are all interconnected by the Turbo architecture. The result is that the customer's 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.

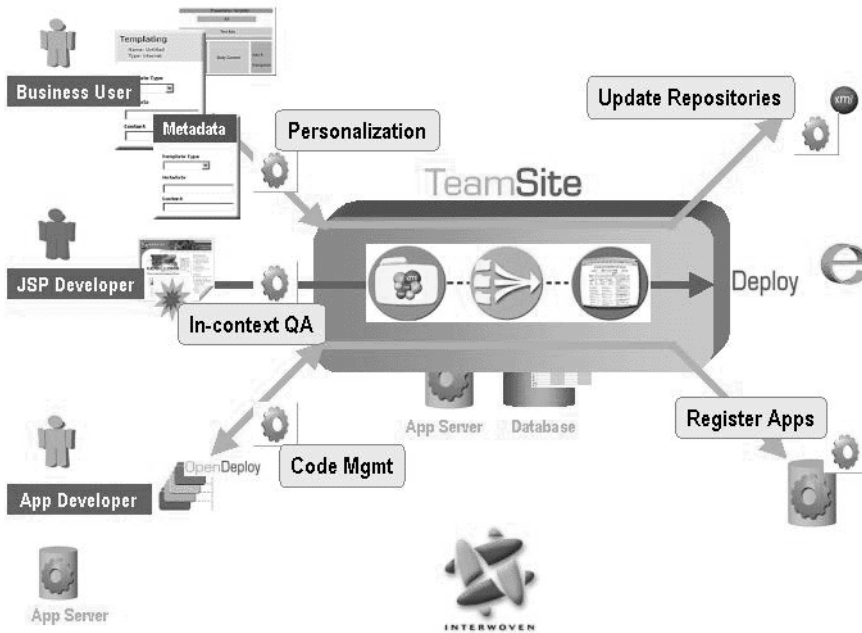


### *Application Server Integration*

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*<sup>™</sup> 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*.



INTERWOVEN



## Chapter 2

# Introduction

---

## Overview

Interwoven Turbo 2.0 for IBM® WebSphere® Personalization (subsequently, also referred to as the *Turbo for WebSphere Personalization*) creates a seamless interface between the Interwoven TeamSite Templating application and the IBM WebSphere Personalization database. TeamSite Templating provides a flexible way to create personalized content through easy-to-use templates and then display that content in different ways based on various presentation templates. WebSphere Personalization allows content contributors to personalize Web pages for individual visitors using profile data collected on the sites or retrieved from other business systems. With Turbo for WebSphere Personalization, you can create and modify content in the WebSphere Personalization database through TeamSite Templating.

## Prerequisite Software

### Development Environment

#### Database Server Host

IBM DB2 6.1 or later, or other RDBMS software (For a full list of supported databases, refer to the corresponding section of the IBM Web site at [http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx\\_aas.htm](http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx_aas.htm).)

#### WebSphere Server Host

- Web server for WebSphere Application Server (For a full list of supported Web servers, refer to the corresponding section of the IBM Web site at [http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx\\_aas.htm](http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx_aas.htm).)

- WebSphere Application Server, Advanced or Enterprise Edition 3.5.3 (version 3.5 plus fixpack3)
- efix for fixpack3: pq46019.jar
- WebSphere Personalization 3.5.2.2 (version 3.5.2 plus efix 2)
- Interwoven Turbo 2.0 for IBM WebSphere Application Server (files for WebSphere Application Server host)
- Interwoven Turbo 2.0 for IBM WebSphere Personalization (files for WebSphere Application Server host)

#### **TeamSite Server Host**

- Web server for TeamSite (For a full list of supported Web servers, refer to the *TeamSite Administration Guide*.)
- TeamSite 4.5.1
- TeamSite Templating 4.5.1
- OpenDeploy® 4.5.1
- DataDeploy™ 4.5.1
- Interwoven Turbo 2.0 for IBM WebSphere Application Server (files for TeamSite host)
- Interwoven Turbo 2.0 for IBM WebSphere Personalization (files for TeamSite server and OpenDeploy/DataDeploy client)

## **Production Environment**

#### **Database Server Host**

IBM DB2 6.1 or later, or other RDBMS software (For a full list of supported Web servers, refer to the corresponding section of the IBM Web site at

[http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx\\_aas.htm](http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx_aas.htm).)

### **WebSphere Server Host**

- Web server for WebSphere Application Server (For a full list of supported Web servers, refer to the corresponding section of the IBM Web site at [http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx\\_aas.htm](http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx_aas.htm).)
- WebSphere Application Server, Advanced or Enterprise Edition 3.5.3 (version 3.5 plus fixpack3)
- efix for fixpack3: `pq46019.jar`
- WebSphere Personalization 3.5.2.2 (version 3.5.2 plus efix 2)
- OpenDeploy 4.5.1
- DataDeploy 4.5.1
- Interwoven Turbo 2.0 for IBM WebSphere Personalization (files for OpenDeploy/DataDeploy server)

### **Client**

WebSphere Studio Advanced Edition 3.5.2

## **Prerequisite Hardware**

### **TeamSite Host Configuration**

For detailed information on TeamSite server host hardware requirements, refer to the *TeamSite Administration Guide*.

### **WebSphere Application Server Host Configuration**

For detailed information on WebSphere Application Server hardware requirements, refer to the corresponding section of IBM's Web site at

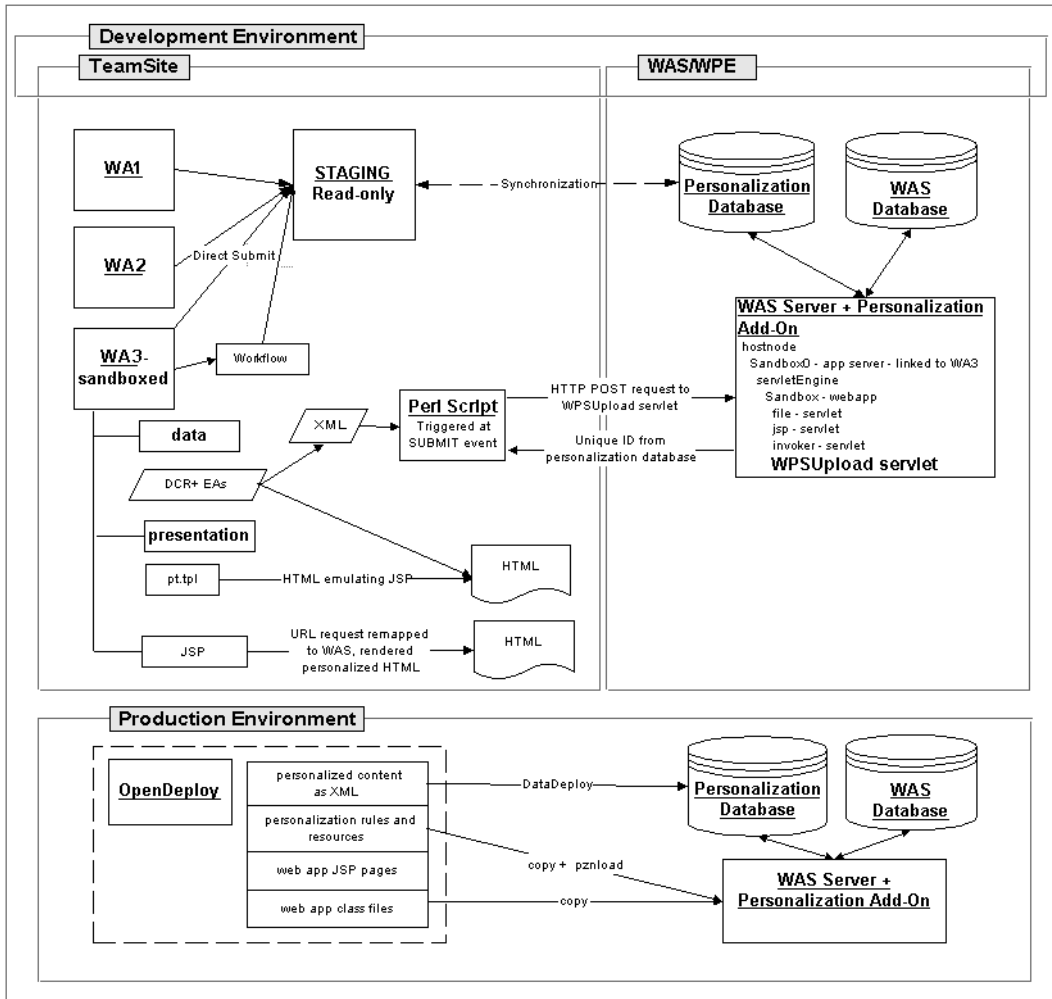
[http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx\\_aas.htm](http://www-4.ibm.com/software/webservers/appserv/doc/latest/idx_aas.htm).

## Product Architecture

The solution architecture of Turbo for WebSphere Personalization can be divided into five major functional areas:

- Data Capture
- Data Preview (phase 1)
- Data Transfer
- Data Preview (phase 2)
- Migration to Production

The diagram below shows the relationship between the TeamSite server and the WebSphere Application Server host in the development and production environments.



*Turbo 2.0 for IBM WebSphere Personalization Solution Architecture*

## Data Capture

Content contribution is performed entirely by the standard TeamSite Templating data capture subsystem. After the contributor enters data into data capture templates (DCTs), TeamSite saves the data as XML files called data content records (DCRs). Content metadata is saved as extended attributes of the DCRs. For more detailed information on the data capture subsystem, refer to the *TeamSite Templating User's Guide* and the *TeamSite Templating Developer's Guide*.

The sample DCTs and metadata capture templates supplied with this product have been created for entering content for the `articles` and `companynews` data types of the `YourCo Toys` sample personalization application.

## Data Preview (phase 1)

The user can preview content in simple HTML format. A presentation template gathers the data from the DCR and generates the HTML page that emulates the look and feel of the personalized JSP page.

## Data Transfer

Submitting a DCR to the TeamSite staging area initiates the data transfer from the TeamSite XML repository to the personalization database. In response to a submit operation the TeamSite server fires the `iwatsub` command trigger. This command trigger executes a Perl script (`iwwpre_submit.ipl`) that gathers DCR data and extended attributes, combining them into a single XML document according to the `contribution` servlet specification. The Perl script uses special presentation templates (`.tpl` files) called DCR parsers to obtain DCR data and extended attributes. DCR parsers build XML tags, which `iwwpe_submit.ipl` combines into an XML file. Then `iwwpe_submit.ipl` invokes the `contribution` servlet through an HTTP POST, providing the servlet with an XML file as the input parameter.

The `contribution` servlet uses resource mapping files to determine the appropriate resource classes to call and it maps resource class properties to corresponding tags from the input XML file. The `contribution` servlet invokes the appropriate action method of resource manager to perform a database transaction. The servlet then constructs a return XML file, which the Perl script intercepts and analyzes to determine the result of the whole data transfer process. The Perl script (`iwwpe_submit.ipl`) extracts the `primary_key` tag value from the return XML file and attaches this value as the `primary_key` extended attribute to the DCR in the workarea if the `primary_key` extended attribute is not yet present, or if the `primary_key` is different from the one returned by

the servlet. Depending on the data in the return XML file, the Perl script either synchronizes the contributor's workarea with the staging area (in case of success), or it performs a staging rollback, restoring the staging area to its state before the submit operation (in case of failure).

There are several important points to understand about the data transfer process:

- To initiate the process, it is not necessary to submit a DCR directly to the staging area. Contributors may also submit a DCR to a workflow. When the DCR eventually reaches the staging area, the submit event will trigger the data transfer process.
- The personalization database is synchronized with content in the staging area, not in the workareas.
- The solution uses the primary key value from the personalization table to keep track of the relationship between the DCR and its corresponding database table row. This value is attached to the DCR as the `primary_key` extended attribute.
- To submit content to the personalization database, the workarea must be properly sandboxed and contain at least the personalization resource classes. The `contribution` servlet uses these classes to perform database transactions, so it must have access to them (via the classpath). Turbo for WebSphere Personalization registers the `contribution` servlet in the sandbox linked to a contributor's workarea, so the workarea must contain resource classes for the servlet to use. This is not the only solution possible, but it is the one the product automates.

## Data Preview (phase 2)

Turbo for WebSphere Personalization allows the content contributor to preview content through personalized JSP pages. The user may navigate through the personalized application from within the workarea using the TeamSite UI to preview/QA the personalized content. Turbo for WebSphere Personalization uses the `YourCo Toys` sample application supplied by WebSphere Studio Advanced Edition to illustrate this feature.

## Migration to Production

After the personalized application and database content are ready for migration to a staging or production server, the Interwoven OpenDeploy application performs the transfer. Turbo for WebSphere Personalization uses DataDeploy to extract the content from the database into XML format on the development server. It then loads the content from XML to the database on the staging or production server. Turbo for WebSphere Personalization contains OpenDeploy and DataDeploy configuration files, scripts, and utilities, which allow the user to deploy all parts of the personalized Web application to production, including JSP pages, class files, database content, personalization rules, and resources.



## Chapter 3

# Installation and Configuration

---

## Overview

This chapter describes how to install and configure Turbo 2.0 for IBM WebSphere Personalization. The installation consists of three parts:

- Installing Turbo for WebSphere Personalization files on the WebSphere Application Server host.
- Updating the sample database.
- Installing Turbo for WebSphere Personalization files on the TeamSite server host.

After completing these three steps, read Chapter 4 to learn how to publish the `YourCo Toys` sample personalization application files into the selected workarea, so contributors can create content for this application. The `YourCo Toys` sample application is distributed with IBM WebSphere Studio Advanced Edition 3.5.2, and is meant to familiarize content contributors with the features and functions of WebSphere Personalization.

## Installing Turbo for WebSphere Personalization on the WebSphere Application Server Host

### Terms

The following terms will be used throughout this chapter:

- *was-root*: a variable referring to the WebSphere Application Server installation root. (For example, `C:\WebSphere\AppServer`)
- *iw-home*: a variable referring to the TeamSite home. (For example, `C:\iw-home`)

## Before You Begin

Familiarize yourself with the features and terminology of Turbo 2.0 for IBM WebSphere Application Server, because Turbo for WebSphere Personalization is an add-on to the WebSphere Application Server. Then go through the following checklist before you install Turbo for WebSphere Personalization on the WebSphere Application Server host:

- Make sure the WebSphere Application Server (with all necessary patches) and the WebSphere Personalization add-on are installed on the host. It is assumed that the WebSphere Application Server and its Web server are on the same host.
- Ensure that the WebSphere Application Server administration host has been started and that the “Default Server” is running properly.
- In Windows, ensure that the TeamSite IFS is mapped to the WebSphere Application Server host as a network drive. Enter **z** as the drive letter for the connection and specify `\\ts_host_name\IWServer` as the connection directory.

In Solaris, ensure that the TeamSite `/iwmnt` directory is mounted to the WebSphere Application Server host.

- If Turbo for WebSphere Application Server is already installed and the sandbox pools exist, destroy lightweight and heavyweight sandbox pools. You will recreate sandbox pools manually after you configure Turbo for WebSphere Personalization with the contribution servlet.
- If Turbo for WebSphere Application Server is not installed, install it following the instructions in the *Interwoven Turbo Product Guide for IBM WebSphere Application Server*.
- Ensure that `WAS_HOME` and `JAVA_HOME` system environment variables (`$WAS_HOME` and `$JAVA_HOME` in Solaris) are set correctly. For correct settings, see the *Interwoven Turbo Product Guide for IBM WebSphere Application Server*.

## Installation

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

1. In Windows, log in as Administrator and run the `install.bat` batch file located in the `install/websphere` directory:

```
C:\>cd install\websphere
C:\install\websphere>install.bat
```

In Solaris, log in as `root` and run the `install.sh` script located in the `install/websphere` directory:

```
#cd install/websphere
#./install.sh
```

If the script is not executable, issue the following UNIX `chmod` command:

```
#chmod +x ./install.sh
```

In both Windows and Solaris, follow the instructions below in response to the prompts shown:

2. Install WebSphere Application Server Component (y/n)?

If Turbo for WebSphere Application Server is already installed, type `n`. If it is not installed, type `y` or press **Enter**.

If you choose not to install it, the installation program will verify its presence by examining the WebSphere Application Server node configuration to detect the `Repository` servlet. If the program detects the servlet, it will proceed to the next step. If not, it will terminate.

If you choose to install it, the program will proceed to the next step.

3. Install EIP Component (y/n)?

Select the appropriate answer.

4. Install WebSphere Personalization Component (y/n)?

Type **y** or press **Enter**.

5. WebSphere Installation Path [C:\WebSphere\AppServer]:

Enter the WebSphere Application Server installation root path or press **Enter** to accept the default path. This directory is also known as the *WAS\_HOME* environment variable (*\$WAS\_HOME* in Solaris) or *WAS-root*.

The installation program copies necessary files into the appropriate locations in *WAS-root* and updates the *iwvs.properties* file located in the *WAS-root\properties* directory.

## Post-Installation Checks

Verify that the following files were created in *WAS-root*:

Directory	File	File Description
<i>WAS-root\iwvs\example</i>	<i>pzn_config.xml</i>	XML configuration file that allows the sandbox to register content contribution servlet, along with Error Reporter, File, JSP, and Auto-Invoker servlets.
<i>WAS-root\personalization\resourcemappings</i>	<i>Articles.xml</i>	XML configuration file that allows the content contribution servlet to map DCR fields to article resource class properties.
<i>WAS-root\personalization\resourcemappings</i>	<i>Companynews.xml</i>	XML configuration file that allows the content contribution servlet to map DCR fields to companynews resource class properties.
<i>WAS-root\properties</i>	<i>iwvs.properties.iwwpe-bak</i>	Backup file of <i>iwvs.properties</i> .

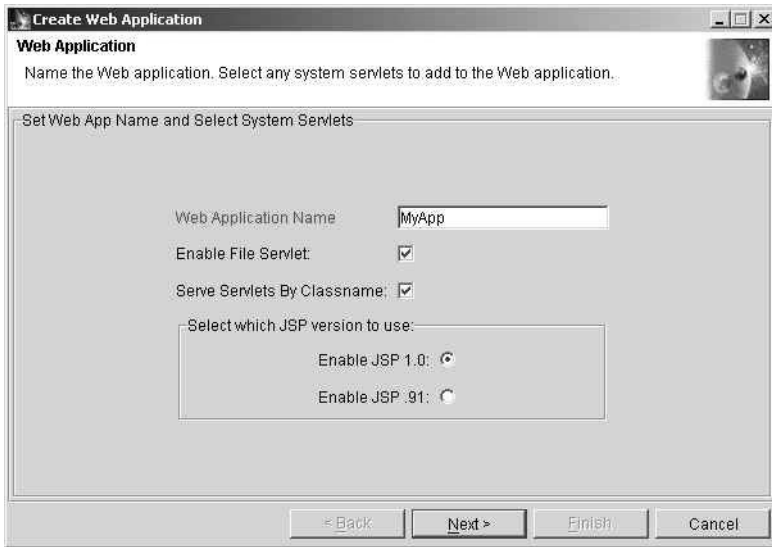
For more information about these files and to find out how to customize some of them, refer to chapters 5 and 6 of this manual.

## Updating the SAMPLE Database

Turbo for WebSphere Personalization uses a sample personalization application called `YourCo Toys`, which is distributed with IBM WebSphere Studio Advanced Edition 3.5.2. This sample application contains JSP pages, servlets, resources, and rules that demonstrate the features of IBM WebSphere Personalization. The `YourCo Toys` sample relies on the SAMPLE database, which you must configure before running the application. After you install Turbo for WebSphere Personalization on the TeamSite server and publish the `YourCo Toys` sample application files into the workarea, contributors can create content for the sample application.

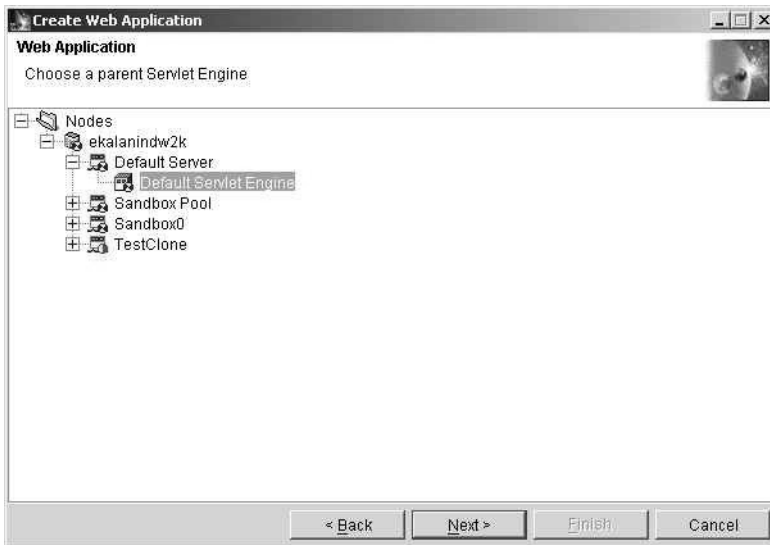
To configure the `YourCo Toys` personalization database:

1. On the WebSphere Administration Console, click the **Wizard** button on the toolbar, and select **Create Web Application** from the pulldown menu. This will initiate a series of dialog boxes. After you enter or accept information in a dialog box, click **Next** to continue.
2. In the first dialog box, enter `myApp` in the **Web Application Name** field and enable the `File`, `Auto-Invoker`, and `JSP 1.0` servlets. (Check **Serve Servlets by Classname** to enable the `Auto-Invoker` servlet.)



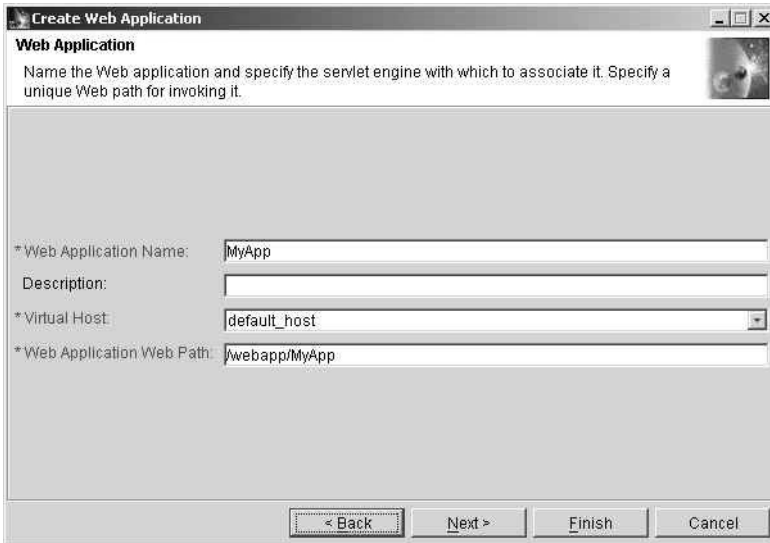
*Create Web Application Dialog Box — entering a Web application name.*

3. In the second dialog box, select `Default Servlet Engine` (in the `Default Server` directory) to be the parent servlet engine for your Web application.



*Create Web Application Dialog Box — selecting a parent servlet engine.*

4. In the third dialog box, accept `default_host` in the **Virtual Host** field and `/webapp/MyApp` in the **Web Application Web Path** field.



**Create Web Application**

**Web Application**

Name the Web application and specify the servlet engine with which to associate it. Specify a unique Web path for invoking it.

\* Web Application Name: MyApp

Description:

\* Virtual Host: default\_host

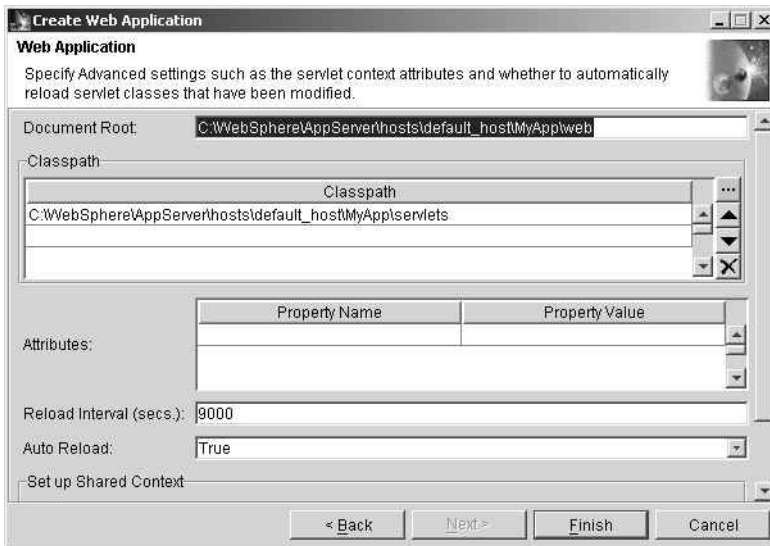
\* Web Application Web Path: /webapp/MyApp

< Back   Next >   Finish   Cancel

*Create Web Application Dialog Box — specifying a Web path.*



5. In the fourth dialog box, accept `WAS-root\hosts\default_host\MyApp\web` as the document root and `WAS-root\hosts\default_host\MyApp\servlets` as the classpath. Click **Finish**.



*Create Web Application Dialog Box — specifying advanced settings.*

6. Create directories for the application docroot and servlets on your file system.
7. Run WebSphere Studio.
8. Within WebSphere Studio, publish `StudioSamples` and `Personalization` projects into the “MyApp” Web application using the archives located in the `studio-root\samples\AppServer3.5` directory. For information on opening sample archives in Studio and publishing them into the WebSphere Application Server web app, refer to the “Working with Studio Samples” chapter in the *IBM WebSphere Studio Guide*.
9. Configure the `SAMPLE` database according to the instructions in the *IBM WebSphere Studio Guide*. When the database is configured, you will be able to run all Studio sample applications, including the `YourCo Toys` personalization application.

10. Log in to the `YourCo Toys` application as `john` (using `john` as the password, as well) to make sure the application was published correctly and is fully functioning.

To find other valid login names and passwords for the `YourCo Toys` application, select the `Userid` and `Password` columns from the `Personnel` table in the `SAMPLE` database.

## Installing Turbo for WebSphere Personalization on the TeamSite Server

### Before You Begin

Make note of the following information, which the installation program will request:

- The branch name and one workarea name that you will want to configure. (If you enter the wrong branch name or workarea name, the installation program will be unable to configure TeamSite Templating or the initial workarea, and you will have to complete those tasks manually.)
- The outgoing mail server host name.
- The WebSphere Application Server host name.
- The Web server port number for the WebSphere Application Server host. (The WebSphere Application Server and its Web server are assumed to be on the same host.)

Make sure that TeamSite Templating is installed and functioning, and that the following configuration files are present:

- `iw-home\local\config\datacapture.cfg`
- `iw-home\local\config\metadata-rules.cfg`
- `iw-home\local\config\templating.cfg`

**Note:** For information on obtaining and configuring the `metadata-rules.cfg` file, see the *Interwoven TeamXpress for Multiplatforms, WebSphere Edition Administration Guide*.

### Installation

1. In Windows, log in as Administrator and run the `install.bat` batch file located in the `install\teamsite` directory:

```
C:\>cd install\teamsite
C:\install\teamsite>install.bat
```

In Solaris, log in as `root` and run the `install.sh` script located in the `install/teamsite` directory:

```
#cd install/teamsite
#./install.sh
```

If the script does not have permission to execute, issue the following UNIX `chmod` command:

```
#chmod +x ./install.sh
```

In both Windows and Solaris, follow the instructions below in response to the following prompts:

2. Install WebSphere Application Server Component (y/n)?

If Turbo for WebSphere Application Server is already installed, type `n`. If it is not installed, type `y` or press **Enter**.

If you choose not to install Turbo for WebSphere Application Server, the installation program will verify that it is present. If it is not present, the installation program will terminate. If it is present, the program will continue to the next step.

If you choose to install Turbo for WebSphere Application Server, the program will install and continue to the next step.

3. Install EIP Component (y/n)?

Select the appropriate answer.

4. Install WebSphere Personalization Component (y/n)?

Type `y` or press **Enter**.

5. TeamSite Installation Path [C:\iw-home]:

Enter the TeamSite installation root path or press **Enter** to accept the default path. This directory is also known as the `TS_HOME` environment variable (`$TS_HOME` in Solaris) or `iw-home`.

The installation program copies necessary files into their appropriate locations in `iw-home`.

6. Enter your outgoing mail server host name:

Enter the name. For example: `smtp.mycompany.com`.

The installation program uses this value when it creates the Turbo for WebSphere Personalization configuration file `iwwpe.cfg` in the `iw-home\wpe\config` directory. To learn about customizing this file, see “Customizing the `iwwpe.cfg` Configuration File” on page 69.

7. Enter the TeamSite branch name you want to configure:

Enter the branch name. This is a required value and it is case-sensitive.

8. Enter the name of the initial workarea you want to configure:

Enter your designated initial workarea. This is a required value and it is case-sensitive.

**Note:** If you enter a nonexistent branch or workarea name, the installation program will not be able to configure TeamSite Templating to support Turbo for WebSphere Personalization and the initial workarea. You will have to configure TeamSite Templating and the workarea manually.

The installation program copies templating files into their appropriate locations in the workarea. It then updates the templating configuration files so that TeamSite Templating supports Turbo for WebSphere Personalization. The program creates backup files during this process. The newly created files are listed below:

Directory	File	Backup File Name
<code>iw-home\local\config</code>	<code>datacapture.cfg</code>	<code>datacapture.cfg.iwwpe-bak</code>
<code>iw-home\local\config</code>	<code>metadata-rules.cfg</code>	<code>metadata-rules.cfg.iwwpe-bak</code>
<code>iw-home\local\config</code>	<code>templating.cfg</code>	<code>templating.cfg.iwwpe-bak</code>

Do not attempt to change the backup files if you wish to preserve your previous configuration.

## Post-Installation Checks

- Verify that the following files were created in *iw-home*:

Directory	File	File Description
<i>iw-home</i> \wpe\bin	iwwpe_submit.ipl	Perl script invoked by <i>iwatsub</i> command trigger. Initiates the content transfer into the personalization database.
<i>iw-home</i> \wpe\bin	iwwp_parse_dcr_articles.tpl	DCR parser to create input XML file out of <i>articles</i> DCR data and metadata.
<i>iw-home</i> \wpe\bin	iwwp_parse_dcr_companynews.tpl	DCR parser to create input XML file out of <i>companynews</i> DCR data and metadata.
<i>iw-home</i> \wpe\config	email_map.cfg	Configuration file that allows mapping between TeamSite users and their email addresses.
<i>iw-home</i> \wpe\config	iwwpe.cfg	XML configuration file for Turbo for WebSphere Personalization.
<i>iw-home</i> \local\config	datacapture.cfg	XML configuration file that defines two rulesets for capturing metadata.
<i>iw-home</i> \local\config	metadata-rules.cfg	XML configuration file that maps vpaths to a rulesets.
<i>iw-home</i> \local\config	templating.cfg	XML configuration file that defines data categories, types, and presentation templates that might be used for each data type. For more information about TeamSite Templating configuration files, refer to the <i>TeamSite Templating Developer's Guide</i> .

- Verify that the selected workarea under the selected branch contains the following files and directories:

Directory	File	File Description
wa	pzn_config.xml	XML configuration file that allows the heavyweight sandbox to register the content contribution servlet, along with Error Reporter, File, JSP, and Auto-Invoker servlets.
wa\templatedata\personalization\articles	datacapture.cfg	XML configuration file that defines data capture requirements for articles data type.
wa\templatedata\personalization\companynews	datacapture.cfg	XML configuration file that defines data capture requirements for companynews data type.
wa\templatedata\personalization\articles\presentation	articles.tpl	Presentation template, which can generate the HTML page that emulates JSP for articles data type.
wa\templatedata\personalization\companynews\presentation	companynews.tpl	Presentation template, which can generate the HTML page that emulates JSP for companynews data type.
wa\templatedata\personalization\articles\data	--	Directory that will contain DCRs of articles data type.
wa\templatedata\personalization\companynews\data	--	Directory that will contain DCRs of companynews data type.
wa\templatedata\personalization\articles\track_files	--	Directory that is marked private and contains track files for articles DCRs. Do not attempt to change the track files manually.

Directory	File	File Description
wa\templatedata\ personalization\ companynews\track_files	--	Directory that is marked private and contains track files for companynews DCRs. Do not attempt to change the track files manually.

For more information about these files and how to customize the product, refer to chapter 5 of this manual.

- If you need to configure the workarea manually, copy all files and directories from the `install\teamsite\workarea` directory to the root of your workarea using your operating system commands or the TeamSite user interface. If you need to configure TeamSite Templating manually, refer to *Interwoven TeamSite Templating Developer's Guide* for instructions and use the following files from `install\teamsite\iw-home\local\config`:

- `datacapture.cfg.iwwpe`
- `templating.cfg.iwwpe`

You will need to create the `metadata-rules.cfg` file yourself.

- Verify that you can create/modify/delete DCRs for `articles` and `companynews` data types. Make sure you can assign correct metadata to the DCRs.
- Verify that you can successfully allocate a sandbox (lightweight or heavyweight) for the initial workarea. To allocate a heavyweight sandbox that can support Turbo for WebSphere Personalization, customize and use the `pzn_config.xml` file located in the root of the workarea. On the WebSphere Application Server host, use the WebSphere Administration console to make sure that the allocated sandbox contains path to the `contentContribution.jar` in its classpath.

For Turbo for WebSphere Personalization to work with the `YourCo Toys` sample application, allocate either a lightweight or a heavyweight sandbox to the workarea.

## Installing and Configuring the iwatsub Command Trigger

To ensure that the `iwwe_submit.ipl` Perl script automatically executes at every submit event, you must configure the `iwatsub` command trigger on the TeamSite server host. To do this, refer to the *TeamSite Command-Line Tools Manual* for platform-specific information about how to configure command triggers.

- In Windows, make sure that the user account that starts the `iwatsub` service is present in the `master.uid` file located in the `iw-home\conf\roles` directory. For example, if the `LocalSystem` account starts the service, add **SYSTEM** into the `master.uid` file; if the `.\Administrator` account starts the service, add **Administrator** into the `master.uid` file.
- In Solaris, make sure that the user account that runs the `$IW_HOME/wpe/bin/iwwe_submit.ipl` script is present in the `master.uid` file located in `iw-home/conf/roles` directory. For example, if `root` runs the script, add `root` into the `master.uid` file.

Specific values for the `Parameters` registry key and example values for the `iw-home/local/iwlocal.cfg` file are given below.

- In Windows:
  - **Application** value = `iw-home\iw-perl\bin\iwperl.exe`
  - **AppParameters** value =  
`iw-home\bin\iwatsub.ipl iw-home\wpe\bin\iwwe_submit.ipl`
  - **AppDirectory** value = `iw-home\wpe`

- In Solaris:

```
#more /usr/iw-home/local/iwlocal.cfg
#IWATPROG USER  COMMAND LINE (default loc. in $IW_HOME/local)
iwatsub  root  $IW_HOME/wpe/bin/iwwe_submit.ipl
```



# Uninstalling Turbo for WebSphere Personalization on the WebSphere Application Server Host

The uninstallation program can remove the Turbo for WebSphere Personalization alone, or all components together. It cannot remove Turbo for WebSphere Application Server alone, however, because Turbo for WebSphere Personalization cannot function without it.

## Before You Begin

- Ensure that the WebSphere Application Server administration server has been started and that the default server is running properly.
- If you plan to remove only Turbo for WebSphere Personalization, make sure that all sandboxes are checked in (not linked to TeamSite workareas). You may also opt to destroy the sandboxes manually before uninstallation using the `sandBoxAdmin.bat` utility (`sandBoxAdmin.sh` in Solaris). The uninstallation program will reconfigure Turbo for WebSphere Application Server to remove support for Turbo for WebSphere Personalization, but it cannot dynamically reconfigure existing sandboxes. If you leave existing sandboxes intact, the uninstallation program will offer to destroy them for you.

## Uninstallation

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

1. In Windows, log in as `Administrator` and run the `uninstall.bat` batch file located in the `install\websphere` directory:

```
C:\>cd install\websphere
C:\install\websphere>uninstall.bat
```

In Solaris, log in as `root` and run the `uninstall.sh` script located in the `install/websphere` directory:

```
#cd install/websphere
#./uninstall.sh
```



If the script is not executable, issue the following UNIX `chmod` command:

```
#chmod +x ./uninstall.sh
```

In both Windows and Solaris, follow the instructions below in response to the following prompts:

2. Uninstall all components (y/n)? [y]:

To remove everything, type **y** or press **Enter**. To remove Turbo for WebSphere Personalization only, type **n**. The uninstallation program cannot remove Turbo for WebSphere Application Server alone without removing the other components.

If you remove all components, the uninstallation program will destroy the sandbox pools, stop and remove the `Repository` servlet, and delete Turbo for WebSphere Application Server, Turbo for WebSphere Personalization, and Turbo for Enterprise Information Portal® and Content Management® files from `WAS-root`. The uninstallation process will end here.

If you remove only Turbo for WebSphere Personalization, the uninstallation program restores the properties file (`iwws.properties`) from the backup file (`iwws.properties.iwwpe-bak`). The program then restarts the `Repository` servlet (so that changed properties can take effect) and removes the individual Turbo for WebSphere Personalization files from their installation directories in `WAS-root`. The uninstallation program then continues:

3. Uninstall EIP component (y/n)? [y]:

Choose the appropriate answer.

4. Uninstall WebSphere Personalization component (y/n)? [y]:

Type **y** or press **Enter**.

5. Destroy sandboxes (y/n)? [y]:

If you wish to remove the sandboxes, type **y** or press **Enter**. The program removes the sandboxes and uninstallation is complete.

If you wish to preserve the sandboxes, type **n**. You will later have to destroy and recreate sandbox pools manually using the `sandBoxAdmin.bat` utility (`sandBoxAdmin.sh` on Solaris).

# Uninstalling Turbo for WebSphere Personalization on the TeamSite Server

Remember that the uninstallation program is case-sensitive.

1. In Windows, log in as Administrator and run the `uninstall.bat` batch file located in the `install\teamsite` directory:

```
C:\>cd install\teamsite
C:\install\teamsite>uninstall.bat
```

In Solaris, log in as `root` and run the `uninstall.sh` script located in the `teamsite-iwvs-install` directory:

```
#cd install/teamsite
#./uninstall.sh
```

If the script is not executable, issue the following UNIX `chmod` command:

```
#chmod +x ./uninstall.sh
```

In both Windows and Solaris, follow the instructions below in response to the following prompts:

2. Uninstall all components (y/n)? [y]:

To remove everything, type **y** or press **Enter**. To remove Turbo for WebSphere Personalization only, type **n**. The uninstallation program cannot remove Turbo for WebSphere Application Server alone without disabling the other components.

3. Uninstall EIP component (y/n)? [y]:

Choose the appropriate answer.

4. Uninstall WebSphere Personalization component (y/n)? [y]:

Type **y** or press **Enter**.

6. TeamSite Installation Path [C:\iw-home]:

Enter the TeamSite installation root path `iw-home` or press **Enter** to accept the default path.



The program removes the individual Turbo for WebSphere Personalization files from their installation directories in *iw-home*, and then restores the TeamSite Templating configuration files (*except templating.cfg*) from their backup and completes.

## Chapter 4

# Using the Sample Application

---

## Overview

WebSphere Studio Advanced Edition is distributed with a sample application called *YourCo Toys*. Turbo 2.0 for IBM WebSphere Personalization supports the *YourCo Toys* sample application, so content contributors can explore the features and functionality of the integration solution. Before contributors can create personalization content for *YourCo Toys* from the TeamSite workarea, you must first publish at least the sample application's resource classes into the selected workarea.

## Publishing the Sample Application into the Workarea

This section explains how to publish the whole personalization application from WebSphere Studio into the workarea. After completing these steps, you will be able to run the virtualized *YourCo Toys* application from within TeamSite.

Perform the following operations from the client:

1. Install WebSphere Studio or verify that the correct version is already installed. To facilitate the WebSphere Studio publishing process, it is helpful to install a Web server on the same computer. For details about publishing servers, refer to the “Publish Your Work” chapter in *IBM WebSphere Studio Guide*.
2. Verify that you have full access rights to your designated workarea. For your designated workarea, use the initial workarea. These instructions assume that you have mapped TeamSite IFS to your client as the `z:` drive, but you can use other drive letters as well.



**Note:** To publish files from Studio to the workarea, you have to set up publishing targets, which must be valid paths to existing directories. If your TeamSite server host is on a UNIX platform, you must install and configure Samba on that host to be able to share TeamSite IFS.

3. Log in to TeamSite and allocate either a lightweight or a heavyweight sandbox for your workarea.

If you allocate a lightweight sandbox, it will be configured automatically with the necessary servlets, including the `contribution` servlet.

If you allocate a heavyweight sandbox, use the `pzn_config.xml` configuration file located in the root of the workarea. The `pzn_config.xml` file must contain the `contribution` servlet configuration information and other necessary servlet configuration information, which will be loaded into your heavyweight sandbox.

4. Run WebSphere Studio and open the `personalization.wsr` archive, located in the `studio-root\Samples\AppServer3.5` directory.

5. To set up the test publishing stage for the project, go to the WebSphere Studio Publishing panel and click **Properties for Publishing Server** to open the Publishing Server Properties dialog box. Enter the webapp web path of the sandbox that is linked to your workarea. The webapp web path is the alias to your virtual docroot, which is your workarea. It appears in this form:

`/default/main/branch/workarea/wa`

where *branch* is a TeamSite branch name and *wa* is your workarea name. (For a detailed explanation of the importance of the webapp web path in this solution, see “Webapp Web Path” under “Important Notes on Publishing the Sample Application” on page 48.)

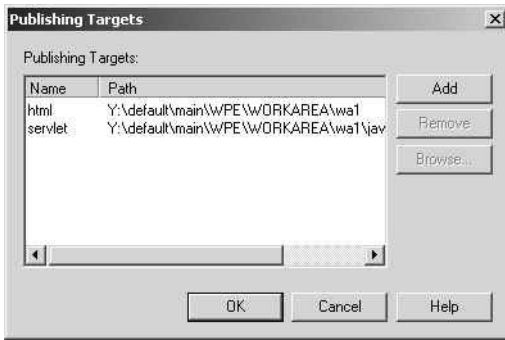


*The Publishing Server Properties Dialog Box*

6. In the Publishing Server Properties dialog box, click **Targets** to open the Publishing Targets dialog box. Set up the following publishing targets:

For html, set up `Y:\default\main\branch\workarea\wa`

For servlet, set up `Y:\default\main\branch\workarea\wa\java`



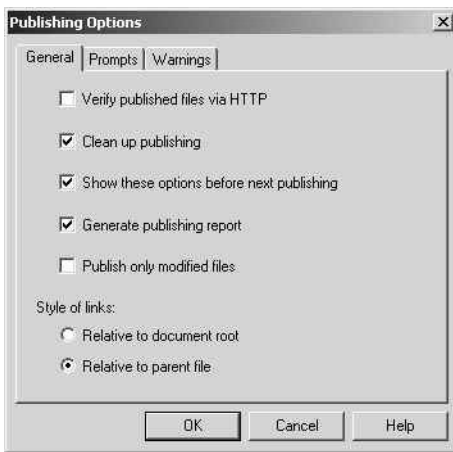
*The Publishing Targets Dialog Box*

7. On the WebSphere Studio Publishing Stage panel, verify that the `servlet` directory has `servlet` as its publishing target.



8. On the WebSphere Studio toolbar, select **Tools > Publishing Options** to open the Publishing Options dialog box. Verify that **style of links** is set to **Relative to parent file**. (For a detailed explanation of the importance of this step, see “Style of Links” under “Important Notes on Publishing the Sample Application” on page 48.)

Check **Generate Publishing Report** to produce a report with which to review the publishing operation results.



*Publishing Options Dialog Box*

9. Publish the `StudioSamples` directory and the `servlet` directory. Review the publishing report to ensure there were no errors during publishing.

It is not necessary to publish the `resources` or the `rules` directories, because the `resources` and `rules` XML files were published into their destination when you created the “MyApp” webapp for configuring the SAMPLE database. (For details, refer to “Updating the SAMPLE Database” on page 27.) The destinations for the `rules` and `resources` directories are `WAS-root\personalization\publishedResources` and `WAS-root\personalization\publishedRules`, respectively. These locations are global for the entire WebSphere Application Server and not virtualized through the TeamSite workarea.

10. Switch to TeamSite and click on the `index.html` file located in the `wa\StudioSamples\Personalization` directory to display the YourCo Toys login screen. Log in as `john` (using `john` as the password, as well) or any other valid user name.

Navigate through the YourCo Toys application to make sure it functions correctly.

## Important Notes on Publishing the Sample Application

### Webapp Web Path

Though WebSphere Studio does not generally require a webapp web path value, you must enter one when using the sample application. The reasons for this are as follows:

The TeamSite proxy server will redirect HTTP requests to the WebSphere Application Server by automatically prepending the webapp web path to every HTTP request that originates from your workarea. Therefore, Uniform Resource Identifiers (URIs) in JSP code that resides in the workarea must not contain the webapp web path. The TeamSite proxy server does not intercept every request, however. Some requests, such as when a servlet redirects to a JSP page, for example, are processed entirely by the WebSphere Application Server and therefore require the correct webapp web path to be prepended to the JSP URI.

Sample applications supplied by WebSphere Studio often use servlet-to-JSP redirects. Sample servlets generated by WebSphere Studio Content Wizard (such as the `Login` servlet from the personalization application) use `class_name.servlet` XML configuration files to find a URI for the redirect. Because these URIs must have the webapp web path prepended to them, you must enter the webapp web path when setting up the publishing stage. WebSphere Studio will automatically prepend it to all URIs in `class_name.servlet` files during publishing, thus allowing the sample application to function correctly from one designated workarea. The `class_name.servlet` files will be unusable across multiple workareas, however. This is a limitation of the current product.

The use of the `class_name.servlet` files for servlet-to-JSP redirects is peculiar to the sample application. It is not J2EE compliant and not recommended when developing your own Web applications.

### Style of Links

You must set **Style of links** to **Relative to parent file**.

If you set **Style of links** to **Relative to document root**, WebSphere Studio will prepend the webapp web path to all URIs found in all JSP pages during publishing. JSP code published with this option becomes unusable across multiple workareas because it will contain a reference to a specific workarea in all URIs.

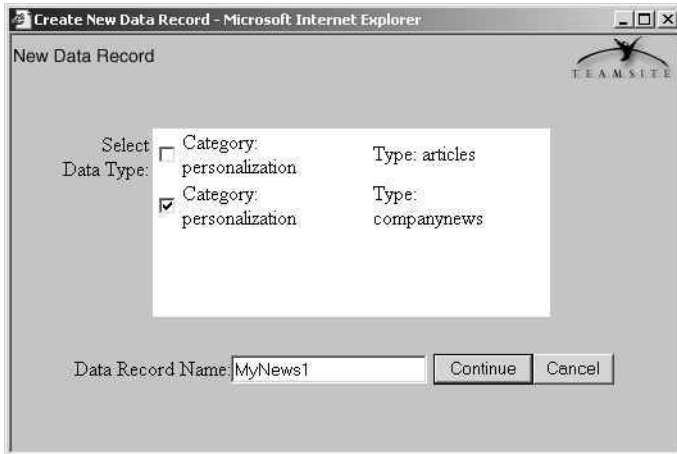
Setting **Style of links** to **Relative to parent file** allows the use of JSP and HTML across multiple workareas, but it does not solve the difficulty with `class_name.servlet` files (See “Webapp Web Path,” directly above), which will contain the webapp web path in URIs regardless of style of links selected. This is a limitation of using the WebSphere Studio sample application from within TeamSite. The samples either must be published in multiple workareas individually or, if the **Get Latest** TeamSite command is used to populate additional workareas, each `class_name.servlet` file must be edited manually to contain a correct URI for a workarea. This also applies to `class_name.servlet` files deployed to production. You will have to adjust the webapp web path manually after deployment.

## Creating Content for the Sample Application through TeamSite Templating

For details on the use of TeamSite Templating, refer to the *TeamSite Templating User's Guide*.

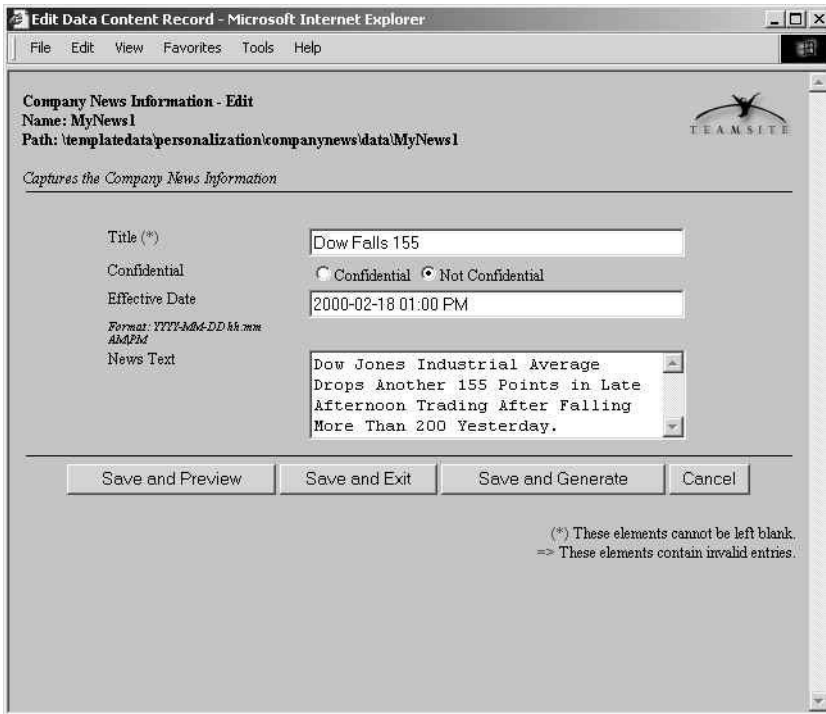
To create personalization content for the `YourCo Toys` application you will use data capture templates in your workarea.

1. Log in to TeamSite and navigate to the initial workarea. Select **File > New Data Record** from the menu bar.
2. In the Create New Data Record dialog box, select the data category and data type for your new record (for example, `personalization` data category and `companynews` data type). Enter a name for the data record in the **Data Record Name** field. Click **Continue**.



*Create New Data Record Dialog Box*

3. In the data capture template (DCT), enter the news title, effective date, and text into the appropriate fields. For this example, enter **2000-02-18 01:00 PM** in the **Effective Date** field. Click **Save and Exit** or **Save and Preview** when finished.



**Edit Data Content Record - Microsoft Internet Explorer**

File Edit View Favorites Tools Help

**Company News Information - Edit**  
Name: MyNews1  
Path: \templatedata\personalization\companynews\data\MyNews1

*Captures the Company News Information*

Title (\*)

Confidential  Confidential  Not Confidential

Effective Date   
Format: YYYY-MM-DD hh:mm AM/PM

News Text

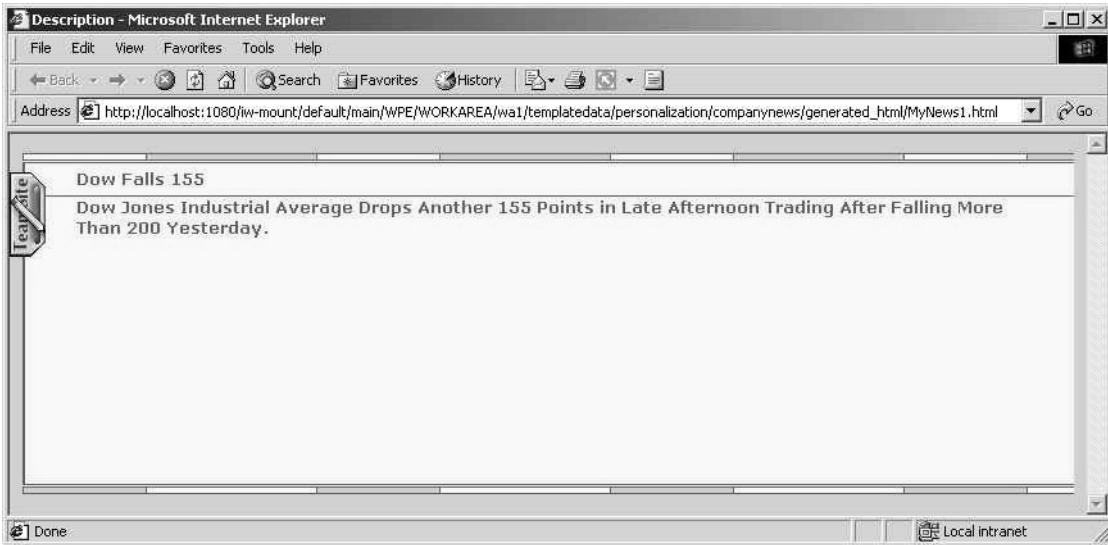
(\*) These elements cannot be left blank.  
=> These elements contain invalid entries.

### *Data Capture Template (DCT)*

If you click **Save and Exit**, the data will be saved as a data content record (DCR) in the TeamSite XML repository.

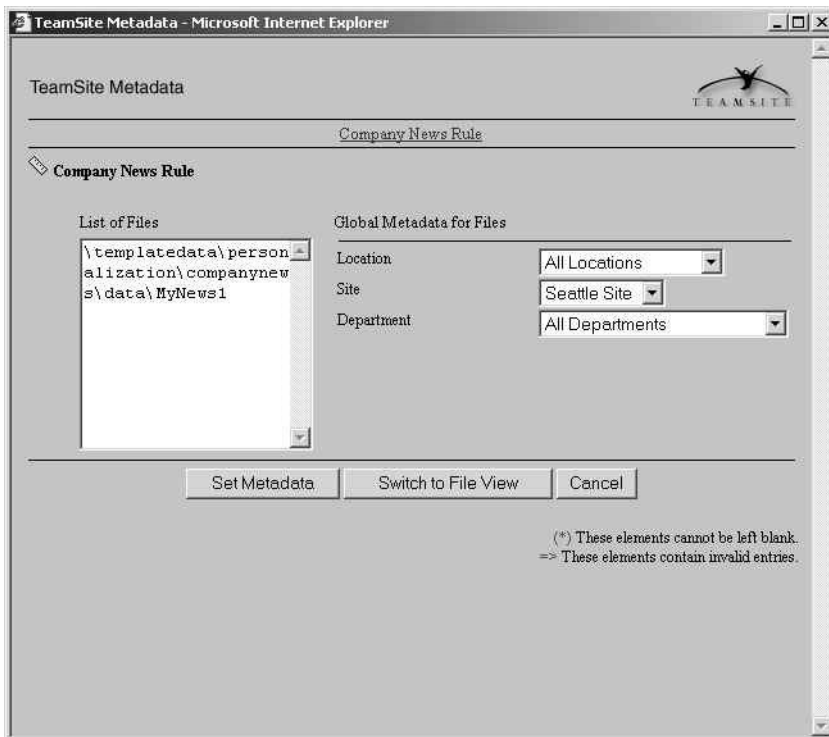
If you click **Save and Preview**, the data will be saved and you will be able to preview your content in HTML format immediately.

The preview allows you to check static content and the look and feel of the page. You will not see personalized content because the preview is an HTML emulation of a personalized JSP page.



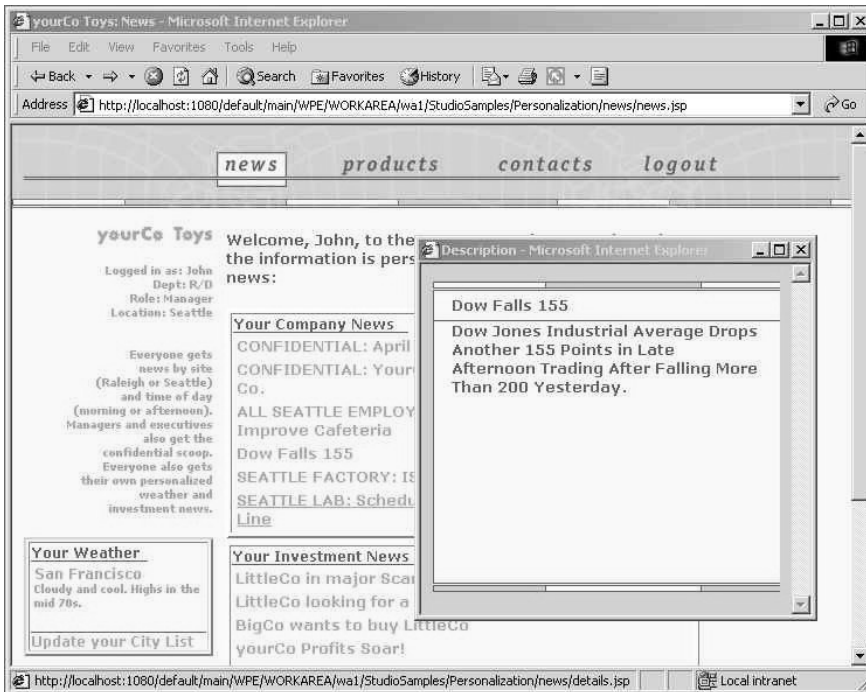
*YourCo Toys News HTML Emulation Page*

4. To assign metadata to your new DCR, select the DCR in the TeamSite GUI and select **File > Set Metadata** from the menu bar to display the metadata capture screen for `companynews` data type. YourCo Toys personalization rules select news content by site. For this example, select **All Locations, Seattle Site**, and **All Departments** from the corresponding pulldown menus. Click **Set Metadata** to save your metadata.



*Metadata Capture Screen*

5. Select **File > Submit Direct** from the TeamSite menu bar to submit your new DCR directly to the staging area, or select **Submit** from the menu or the toolbar to submit to a workflow. The submit operation and data transfer process must finish before you can preview new content using a personalized JSP page. If you submit the DCR to a workflow, the whole workflow and data transfer process must finish before you can preview new content using a JSP page.
6. Run the `YourCo Toys` application from your workarea. Log in as `john` (using `john` as the password, as well) and navigate to the news page to see your new content.



*YourCo Toys News Page*



7. To edit your DCR, select the DCR on the TeamSite GUI and click **Edit File** on the toolbar, or **Edit > Edit Data Record** from the menu bar. On the data capture template, change the news content. (See “Points to Remember” on page 55 for restrictions on modifying the **Title** field.) Save the DCR and submit it to the staging area. After the submit operation and data transfer process are complete, you can view the modified content on the news page.

**Note:** Study the rules of the `YourCo Toys` personalization application to learn how to enter the correct content. To enter the correct content and metadata tags, you must understand how the personalization rules work. If content data does not make sense or if metadata tags contradict the rules, the personalized content may be replicated several times on a single page or it may not appear at all.

8. To delete the personalization content, select **File > Delete Data Record** and submit the deletion. After the submit operation and data transfer process are complete, content is deleted from the personalization database.

## Points to Remember

When creating and modifying content for the `YourCo Toys` application, contributors should keep some important points in mind.

### The Title Value in the `Companynews` and `Articles` Tables

In the `YourCo Toys` sample application, the `companynews` table and the `articles` table define **Title** as the primary key. The use of **Title** as the primary key results in the following limitations, which apply only to `YourCo Toys` content:

- The **Title** value must be unique in every DCR of `companynews` and `articles` data types, because it serves as the primary key value.

When developing your own business solutions, do not create a data model that requires the contributor to enter unique values. TeamSite Templating cannot check for data uniqueness during content entry.

- When entering **Title** data, do not use “:”, “/”, “\” anywhere in the title, “.” at the end of the title, or leading or trailing spaces and tabs. Use of any of these symbols will disable the delete function.



This is because the solution uses the primary key (in this case, **Title**) in a track file that enables it to find and delete a DCR's corresponding database entry. The track file will not be created if it incorporates the symbols listed above, because many operating systems prohibit the use of these symbols in filenames. The track file serves as a backup of the deleted DCR's `primary_key` extended attribute, which links the DCR to its corresponding row in the database.

The solution creates the track file in the format of `DcrName_iw_PrimaryKey` in the `track_files` directory in the workarea after a successful insert transaction and deletes it after a successful delete transaction.

When developing your own personalization databases, define the primary key as a unique meaningless numeric value that is automatically generated (by the database or other mechanisms) for each record during an insert transaction. A numeric value contains none of the restricted characters listed above, which will ensure that the track file can always be created, accessed, and deleted.

- Do not modify the **Title** value after the corresponding database record has been created, because a change at that time will cause the `contribution` servlet to return an error. TeamSite Templating cannot dynamically change write permission on DCR fields, which means contributors are at no time automatically prohibited from modifying the **Title** value in the data capture template. They should, nevertheless, not do so.

This limitation is due to the fact that `companynews` and `articles` resource classes do not have the `setTitle` method defined.

The work-around for this limitation is to define the `setTitle` method for the resource classes, recompile them, and publish them to your workarea. You will then be able to modify the primary key value by changing the **Title** field.

When you develop your own personalization applications, be sure to provide `set` methods for all resource properties that can be modified according to your business logic. `YourCo Toys` resource classes do not provide the `setTitle` method to restrict updates on primary key, but this contradicts business logic because the news title or article title may certainly change.

## A Limitation of the News Page Rules

When you preview the `YourCo Toys` news page, personalized content may appear twice. The explanation for this follows:

The news page uses two personalization rules to select news content: `GetNewsByClearance` and `GetNewsByTime`. The two rules have overlapping parameters. The `GetNewsByClearance` rule selects content according to the `confidential` flag. The `GetNewsByTime` rule uses the `GetMorningNews.act` action and the `GetAfternoonNews.act` action, which select content according to time of the day. Both of these actions have `02-28-2000` hardcoded as the date. News content tagged `confidential` and dated after noon on February 28, 2000 will therefore appear twice.



INTERWOVEN

## Chapter 5

# Customization

---

## Overview

Turbo 2.0 for IBM WebSphere Personalization is distributed with the components that enable the contributor to create content for the `YOURCO_TOYS` sample application. To create content for your own applications, however, you must customize the components of Turbo for WebSphere Personalization. This chapter explains how to customize components to accommodate the requirements of your data model.

You can customize the following components of Turbo for WebSphere Personalization:

- Data categories and data types
- Data capture templates (`datacapture.cfg` files)
- Presentation Templates (`*.tpl` files)
- Metadata rules
- Metadata capture templates
- Resource mapping files
- The `iwwpe.cfg` configuration file
- The `email_map.cfg` configuration file

## Customizing Data Categories and Data Types

TeamSite Templating uses a data storage hierarchy based on data categories and types. The directory structure that supports this hierarchy resides in the workarea and consists of three levels:

- The `templatedata` directory (highest level in the hierarchy, located at the root of the workarea):  
If you would like to change the default name of the directory, refer to the *TeamSite Templating Developer's Guide* for instructions.

- Data categories (represented by directories located in the `templatedata` directory): Turbo for WebSphere Personalization uses the `personalization` data category. Data categories reside in the directory structure, but they also must be listed in the `templating.cfg` configuration file to be available to TeamSite Templating. If you rename the default data category or define additional ones, remember to modify the `templating.cfg` file as well. For details about `templating.cfg` and its sections, refer to the *TeamSite Templating Developer's Guide*.
- Data types: Each data category may contain one or more data types. Turbo for WebSphere Personalization is distributed with two data types under the `personalization` data category: `companynews` and `articles`. Each data type corresponds to a table in the personalization database. If you rename the data types or define additional ones, remember to modify:
  - The `templating.cfg` file.
  - The `iwwpe.cfg` file, which maps data types to parser names and to resource names (For detailed information on this file, refer to page 69.).
  - The `metadata-rules.cfg` file, which maps branches and data types to metadata capture rules.

Below is a section of the `templating.cfg` file that contains out-of-the-box Turbo for WebSphere Personalization definitions:

```
<category name="personalization"> ← default data category name
  <locations>
    <branch vpath-regex="WP" /> ← defined for "WP" branch
  </locations>
  <data-type name="articles"> ← data type name
    <presentation>
      <template name="articles.tpl" extension="html">
        <locations>
          <branch vpath-regex=".*" preview-dir="/">
            <directory dir-regex=".*" />
          </branch>
        </locations>
      </template>
    </presentation>
  </data-type>
```

```

<data-type name="companynews"> ←———— data type name
  <presentation>
    <template name="companynews.tpl" extension="html">
      <locations>
        <branch vpath-regex=".*" preview-dir="/">
          <directory dir-regex=".*" />
        </branch>
      </locations>
    </template>
  </presentation>
</data-type>
</category>

```

## Customizing datacapture.cfg Files

The `datacapture.cfg` file is the XML configuration file that defines a data capture template and drives data capture for a specific data type. It defines input fields, their appearance, default values, data validation rules, and other information that is important for the data entry process. Each data type must have exactly one `datacapture.cfg` file. For more information about the `datacapture.cfg` file and its customizable components, refer to the *TeamSite Templating Developer's Guide*.

Turbo for WebSphere Personalization provides two `datacapture.cfg` files for the YourCo Toys sample application. Each data input field in the sample DCTs contains parameters that describe the corresponding database data type, maximum field capacity, required flag, and a validation rule that prohibits entering illegal data.

Each `datacapture.cfg` file must contain a resource name, which defines a value that exactly matches a resource class name for a corresponding data type. (For the `companynews` data type, for example, the `datacapture.cfg` file contains a resource name with default value `Companynews`, because the corresponding resource class name is `Companynews.class`.) This value is case-sensitive.

If TeamSite Templating is used with a CGI-based user interface, the resource name is hidden to prevent modifications. If TeamSite Templating is used with a Java-based user interface, you must

make the resource name field visible and writable by commenting out the section of the `datacapture.cfg` files shown below. Content contributors should not modify this value.

```
<item name="Resource Name">
  <text required="t">
    <!-- start comment -->
    <!-- This section should be commented out for Java-based TeamSite
Templating user interface -->
    <allowed>
      <not>
        <or>
          <cred role="master"/>
          <cred role="admin"/>
          <cred role="editor"/>
          <cred role="author"/>
        </or>
      </not>
    </allowed>
    <!-- end comment -->
    <default>Companynews</default>
  </text>
</item>
```

Customizing any of the default `datacapture.cfg` files or creating new ones requires knowledge of the database table structure and the resource class name for the corresponding data type. It is best to supply the DCT developer with a DDL script for the database table. The developer has discretion over whether to insert the data in a particular database column through DCTs or through metadata capture templates.

The following schema shows the correspondence between the `companynews` DCT data input fields and the `companynews` table columns.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE datacapture SYSTEM "datacapture4.5.dtd">

<data-capture-requirements type="content">
  <ruleset name="Company News Information">

    <description>
```



Captures the Company News Information  
</description>

<item name="Title"> ← TITLE VARCHAR (80) NOT NULL  
 <database data-type="VARCHAR(80)" />  
 <text size="40" required="t" maxlength="80" />  
</item>

<item name="Confidential"> ← CONFIDENTIAL VARCHAR (1)  
 <database data-type="VARCHAR(1)" />  
 <radio>  
 <option value="1" label="Confidential"/>  
 <option value="0" label="Not Confidential" selected="t"/>  
 </radio>  
</item>

<item name="Effective Date"> ← DATETIME VARCHAR (15)  
 <description>  
 Format: YYYY-MM-DD hh:mm AM|PM  
 </description>  
 <database data-type="VARCHAR(15)" data-format="yyyy-MM-dd hh:mm  
 AM|PM"/>  
 <text size="40" maxlength="19"  
 validation-regex="^[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]  
 [ ]\*[0-1][0-9]:[0-5][0-9] [ ]\*[aApP][mM]\$" />  
</item>

<item name="News Text"> ← CONTENT VARCHAR (1000)  
 <database data-type="VARCHAR(1000)" />  
 <textarea rows="4" cols="34" maxlength="1000" wrap="virtual"/>  
</item>

<item name="Resource Name">  
 <text required="t">  
 <allowed>  
 <not>  
 <or>  
 <cred role="master"/>  
 <cred role="admin"/>  
 <cred role="editor"/>  
 <cred role="author"/>

```
        </or>
        </not>
    </allowed>
    <default>Companynews</default>
</text>
</item>

</ruleset>
</data-capture-requirements>
```

## Customizing Presentation Templates

Presentation templates (PTs) are XML files designed to generate output files using DCR data. PTs may contain custom Interwoven XML tags, HTML, and Perl code. The TeamSite Templating Page Generation Subsystem combines DCR data with presentation templates to generate output files. For detailed information about PTs, refer to the *TeamSite Templating Developer's Guide*.

Turbo for WebSphere Personalization is distributed with two types of PTs for use with the `YourCo Toys` sample application:

- The first type (located in the `presentation` directory for each data type) generates the HTML pages that emulate the look and feel of a JSP page and may be used to preview the content. The HTML tags for these PTs were taken from the `YourCo Toys` application JSP page that displays personalized news. The data for generated output is dynamically inserted from the DCR during page generation. Turbo for WebSphere Personalization is distributed with two such PTs: `articles.tpl` and `companynews.tpl`.
- DCR Parsers (located in the `iw-home/wpe/bin` directory): These generate XML tags that the `iwwpe_submit.ipl` script uses to construct the input XML file for the `contribution` servlet. Turbo for WebSphere Personalization provides DCR parsers for both `companynews` and `articles` data types: `iwwpe_parse_dcr_compnews.tpl` and `iwwpe_parse_dcr_articles.tpl`, respectively.

Below is a format for servlet input XML and an example of generated input XML for the `companynews` data type:

```
<?xml version="1.0" encoding="UTF-8"?>
<input name=resourceName>
  <data>
    <input_field1>data for field 1</input_field1>
    <input_field2>data for field 2</input_field2>
    ...
    <input_fieldN>data for field N</input_fieldN>
  </data>
  <metadata>
    <primary_key>some primary key</primary_key>
  </metadata>
</input>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<input name="Companynews">
  <data>
    <Title>Some title</Title>
    <Confidential>N</Confidential>
    <Effective_Date>200001011400</Effective_Date>
    <News_Text>Some text</News_Text>
    <Location>*</Location>
    <Site>*</Site>
    <Department>*</Department>
  </data>
  <metadata>
    <primary_key>Some title</primary_key>
  </metadata>
</input>
```

The following DCR parser customizations are possible:

- You may name the DCR parser. The correspondence between parser name and data type is recorded in the `iwwpe.cfg` configuration file. (For details, see “Customizing the `iwwpe.cfg` Configuration File” on page 69.) You must record new names for existing or new parsers in this configuration file for the solution to work properly.

- You may name the XML tags the DCR parser generates. The `contribution` servlet uses mapping files (see “Customizing Resource Mapping Files” on page 68) to find the correspondence between XML tags and actual resource properties. The DCR parser therefore can generate tags with any name as long as the mapping file is kept in synch with tag name changes.

When customizing DCR parsers, observe the following restrictions to ensure that all content is inserted into the database:

- Keep the DCR parser in synch with the DCT. If you do not, some required XML tags may be omitted.
- Keep the DCR parser in synch with the corresponding resource mapping file. If you do not, the `contribution` servlet will be unable to map XML tags to resource properties.
- The DCR parser must not generate tags that contain spaces, because the IBM XML parser used by the `contribution` servlet cannot parse such tags. If the tag contains spaces, the `contribution` servlet will consider the whole XML file invalid. (For example, the parser must not generate `Effective Date` as the tag for the **Effective Date** field in the DCR.)

## Customizing Metadata Rules

TeamSite uses the `metadata-rules.cfg` file, located in the `iw-home/local/config` directory, to map vpaths and rule names to the metadata capture rules defined in the `iw-home/local/config/datacapture.cfg`, a metadata capture template file. For detailed information about `metadata-rules.cfg`, refer to the *TeamSite Administration Guide*.

Turbo for WebSphere Personalization provides a `metadata-rules.cfg` file that maps `companynews` data type to `Company News Rule` and `articles` data type to `Articles Rule`.

As mentioned above, the data type name change must be reflected in the `metadata-rules.cfg` file. When adding a new data type that requires metadata capture, insert a new section into the `metadata-rules.cfg` file to reflect the addition.

If you modify the rule name, make sure to also modify the corresponding ruleset name in the metadata capture template file.

## Customizing Metadata Capture Templates

TeamSite uses the `datacapture.cfg` file (metadata capture template, located in the `iw-home/local/config` directory) to define rulesets for metadata capture. For detailed information about this file, refer to the *TeamSite Administration Guide*.

Turbo for WebSphere Personalization provides two rulesets, `Company News Rule` and `Articles Rule`, one for each sample application data type. The rules stipulate that each data input field corresponds to a column in the personalization table and contains a database data type parameter and a set of allowed values.

Customizing any of the provided rules or creating new ones for existing or additional data types requires knowledge of the database table structure and personalization rule queries. It is best to supply the DCT developer with a DDL script for the database table and possible values for each personalization tag. The developer has discretion over whether to insert the data in a particular database column through DCTs or through metadata capture templates.

The following schema shows the correspondence between data input fields for `Company News Rule` and `companynews` table columns:

```
<ruleset name="Company News Rule">
  <item name="Location"> ← FORLOCATION VARCHAR (10)
    <database data-type="VARCHAR(10)" />
    <select>
      <option value="HQ" label="Headquarters" />
      <option value="Factory" label="Factory Location" />
      <option value="Lab" label="Laboratory Location" />
      <option value="*" label="All Locations" />
    </select>
  </item>
  <item name="Site"> ← FORSITE VARCHAR (10)
    <database data-type="VARCHAR(10)" />
    <select>
      <option value="Raleigh" label="Raleigh Site" />
      <option value="Seattle" label="Seattle Site" />
    </select>
  </item>
</ruleset>
```

```
<option value="*" label="All Sites" />
</select>
</item>
<item name="Department">
  <database data-type="VARCHAR(10)" />
  <select>
    <option value="Corp" label="Corporate Office" />
    <option value="R/D" label="Research and Development" />
    <option value="Sales" label="Sales" />
    <option value="Mfg" label="Manufacturing" />
    <option value="*" label="All Departments" />
  </select>
</item>
</ruleset>
```

FORDEPT VARCHAR (10)

## Customizing Resource Mapping Files

Resource mapping files are XML files used by the `contribution` servlet.

The servlet invokes the resource manager class action methods to perform database transactions. The resource classes contain properties (in this case, database table columns) and methods for setting and getting properties values. Resource manager classes contain action methods. The `contribution` servlet finds the appropriate resource, sets resource properties, and then invokes the appropriate action method of the resource manager. The mapping files allow the servlet to determine which resource property corresponds to which input XML tag. It also specifies the property name that corresponds to a primary key.

Turbo for WebSphere Personalization provides two mapping files (`Companynews.xml` and `Articles.xml`), one for each of the YourCo Toys sample application data types. Both are located in the `WAS-root/personalization/resourcemappings` directory. The resource mapping files must be kept in synch with the DCR parsers to ensure that all necessary columns will get values for the database transaction. (See the section on customizing DCR parsers under “Customizing Presentation Templates” on page 64.)

Resource mapping file names are not customizable and must match with the resource class names exactly. Names are case-sensitive.

Below is a `Companynews.xml` resource mapping file (no line breaks should appear in your file):

```
<?xml version="1.0" encoding="UTF-8"?>
<resource>
  <property name="TITLE" input_field="Title" data_type="java.lang.String"
primary_key="yes" />
  <property name="DATETIME" input_field="Effective_Date"
data_type="java.lang.String" />
  <property name="FORLOCATION" input_field="Location"
data_type="java.lang.String" />
  <property name="FORSITE" input_field="Site" data_type="java.lang.String" />
  <property name="FORDEPT" input_field="Department" data_type="java.lang.String"
/>
  <property name="CONFIDENTIAL" input_field="Confidential"
data_type="java.lang.String" />
  <property name="CONTENT" input_field="News_Text" data_type="java.lang.String" /
>
</resource>
```

The Title property corresponds to the Title XML tag and it is a primary key.

## Customizing the `iwwpe.cfg` Configuration File

The `iwwpe.cfg` file is an XML configuration file located in `iw-home/wpe/config` directory and used by the `iwwpe_submit.ip1` trigger script. This file is constructed when the user provides input during product installation on the TeamSite server host. The tag names in the `iwwpe.cfg` file are not customizable, but the tag values provide some flexibility when designing your own solutions.

Below is the `iwwpe.cfg` file. See the “Diagram Key” following for an explanation of each referenced item.

```

<!--#####-->
<!--# iwwpe.cfg - configuration file -->
<!--# Please customize this file to reflect your environment. -->
<!--#####-->

<iwwpe_cfg version='2.0'>

  <key name='MAILSERVER'>
smtp.mycompany.com  ← 1. Outgoing mail server
  </key>
  <key name='EMAIL_MAPPING_FILE'>
C:\iw-home\wpe\config\email_map.cfg ← 2. Email mapping file that maps
TeamSite users to their email
addresses
  </key>
  <key name='WAS_HOST_URL'>
http://somehostname ← 3. URL of the WebSphere
Application Server host
  </key>
  <key name='SERVLET_ALIAS'>
/servlet/contribution ← 4. Web alias for the contribution
servlet
  </key>
  <datatype_parsername datatype='companynews'
parsername='iwwpe_parse_dcr_compnews.tpl' />
  <datatype_parsername datatype='articles'
parsername='iwwpe_parse_dcr_articles.tpl' /> ← 5. Mapping data types to DCR
parsers
  <datatype_resourcename datatype='companynews'
resourcename='Companynews' />
  <datatype_resourcename datatype='articles'
resourcename='Articles' /> ← 6. Mapping data types to
resource name
</iwwpe_cfg>

```



## Diagram Key

1. **Outgoing Mail Server:** The rollback mechanism requires this value to email a failure notification to a workarea owner if the data transfer process fails.
2. **Email Mapping File:** This value is also required for failure notification.
3. **WebSphere Application Server Host URL:** If the WebSphere Application Server port is different from 80, this URL is recorded as `http://somehostname:port`. The `iwwpe_submit.ipl` script requires this value to be able to construct the HTTP request to the `contribution` servlet.
4. **Web Alias for Contribution Servlet:** This alias assumes that Auto-Invoker is used to invoke the servlet. The `iwwpe_submit.ipl` script requires this value to construct the HTTP request to the servlet. The final URL of the request looks similar to this:

```
http://somehostname/default/main/branch/WORKAREA/wa/servlet/  
Registered_Servlet_Name
```

5. **Mapping Data Types to DCR Parsers:** The `iwwpe_submit.ipl` script requires this value to determine which DCR parser to invoke to parse the DCR of this data type. If either value is modified, the change must be reflected here.
6. **Mapping Data Types to Resource Names:** The `iwwpe_submit.ipl` script requires this to be able to determine what resource class name to use when constructing input XML for deleted DCRs of this data type. The script cannot get any DCR values (including “Resource Name”) from a deleted DCR, therefore the data-type-to-resource-name map must be recorded in the configuration file. If either value is modified, the change must be reflected here.

## Customizing the email\_map.cfg Configuration File

The `email_map.cfg` file (located in the `iw-home/wpe/config` directory) provides the mapping between TeamSite user names and their email addresses. The `iwwpe_submit.ipl` script uses the `email_map.cfg` file to obtain the email address to notify a contributor of a data transfer process failure.

It is important to add all TeamSite contributors as well as the TeamSite Administrator to the file. Map the account name that runs the `iwatsub` service (the `iwwpe_submit.ipl` script on Solaris) to the TeamSite administrator's email. For example, map **SYSTEM** (or **root**) to the administrator's email address.

Format the entries as follows:

```
ts_user_name:email_address
```

In Windows, the user name must include the domain name, for example:

```
NTDOMAIN\ts_user_name:email_address
```

Below is an example of the failure notification sent to `Joe_Smith@company.com` by the `iwwpe_submit.ipl` script:

```
To: Joe_Smith@company.com
From: IBM Turbo for WPE Notification Service
Subject: Content Item DB Transaction Failure Notification
```

```
Failure Notice: Workarea: wa_name
                  DCR Name: /default/main/branch/WORKAREA/ \
                           wa_name/templatedata/personalization/ \
                           companynews/data/MyNews
```

A transaction requested on above DCR failed.

===== Failure Reason =====

```
HTTP request to contribution servlet failed.
Contact your System Administrator.
```

=====

If this is due to a IBM Turbo for WPE configuration error  
- please contact your TeamSite or WAS/WPE administrator.

If this is due to a data error  
- please correct the DCR data using a Data Capture Template.

To request this transaction again, submit the corrected  
DCR to the staging area.



INTERWOVEN

## Chapter 6

# Deploying the Sample Application

---

## Overview

This packaged solution provides sample configuration files and scripts to deploy all assets of the `YourCo Toys` sample application from the development server to the staging or production server. The sample configuration files and scripts are fully functional, but require custom environment information. This chapter describes how to customize the sample files and use them for deployment. The samples serve as a base for more sophisticated deployments.

**Note:** The sample deployment configuration files are targeted to the Windows platform. For deployments in the Solaris environment, adapt the files accordingly.

## Before You Begin

- Verify that OpenDeploy and DataDeploy are installed on the TeamSite server host (which serves as the OpenDeploy client) and the WebSphere Application Server staging or production host (which serves as the OpenDeploy server).

The OpenDeploy server host name is noted as `od_server`. The OpenDeploy client host name is noted as `od_client`.

- Verify that DataDeploy and OpenDeploy share the same installation roots on each host.

On the staging or production server, this directory is noted as `od_server_root`. On the development server, this directory is noted as `od_client_root`.

- If `od_server` is a Windows server that uses DB2 as the production database, apply DataDeploy Patch ID 1076 and copy the `db2java.zip` file from the `%DB2HOME%\java` directory on the production database server to the `od_server_root\lib` directory on `od_server` (overwriting the file shipped with DataDeploy).

If *od\_server* is a Solaris server that uses DB2 as the production database, apply DataDeploy Patches ID 1105 and ID 1077 and copy the *db2java.zip* file from the *\$DB2HOME/java* directory on the production database server to the *od\_server\_root/lib* directory on *od\_server* (overwriting the file shipped with DataDeploy).

- If *od\_client* is a Windows server that uses DB2 as the development database, apply DataDeploy Patch ID 1076 on *od\_client*.

If *od\_client* is a Solaris server that uses DB2 as the development database, apply DataDeploy Patches ID 1105 and ID 1077 on *od\_client*.

- The sample deployment files are located in the following directories:
  - OpenDeploy server components are in  
*install/websphere/opendeploy* and  
*install/websphere/pznload*
  - OpenDeploy client components are in  
*install/teamsite/opendeploy*

## Server Components Setup

Copy files from the *install/websphere/opendeploy* and *install/websphere/pznload* directories to the respective destination directories as indicated in the table below.

| File or Directory   | Destination Directory     |
|---|---------------------------|
| <i>install/websphere/opendeploy/bin/iwwpe_dump_load.pl</i><br>(the DNR script)  | <i>od_server_root/bin</i> |
| <i>install/websphere/opendeploy/bin/iwwpe_backup_data.sql</i><br>(SQL script to back up data in the production database to <i>xxx_save</i> tables before loading deployed data. Called by <i>iwwpe_dump_load.pl</i> script with <i>load</i> parameter.) | <i>od_server_root/bin</i> |
| <i>install/websphere/opendeploy/bin/iwwpe_restore_data.sql</i><br>(SQL script that restores original data in the production database from <i>xxx_save</i> tables. Run it manually, if necessary.)   | <i>od_server_root/bin</i> |

| File or Directory  | Destination Directory                                       |
|--|---|
| install/websphere/opendeploy/conf/iwwpe_remote_receive.cfg<br>(configuration file for the OpenDeploy server) | od_server_root/conf   |
| install/websphere/opendeploy/conf/iwwpe_xml_db.cfg<br>(configuration file for XML-to-database deployment)    | od_server_root/conf   |
| websphere/pznload<br>(directory containing configuration files for the pznload utility)                      | WAS_server_root/<br>personalization/<br>publishToProduction |

## Client Components Setup

Copy files from `install/teamsite/opendeploy` to their respective destination directories as indicated in the table below.

| File   | Destination Directory |
|--|-----------------------|
| teamsite/opendeploy/bin/iwwpe_dump_load.ipl<br>(the DNR script)                                  | od_client_root/bin    |
| teamsite/opendeploy/conf/iwwpe_local_send.cfg<br>(configuration file for OpenDeploy client)      | od_client_root/conf   |
| teamsite/opendeploy/conf/iwwpe_db_xml.cfg<br>(configuration file for database-to-XML deployment) | od_client_root/conf   |

## Server Components Configuration

Follow the directions in this section to customize the OpenDeploy and DataDeploy configuration files and other server components, listed below:

- `iwwpe_remote_receive.cfg`
- `iwwpe_xml_db.cfg`
- The `pznload` utility
- `iwwpe_backup_data.sql`
- `iwwpe_restore_data.sql`

**Note:** All directories must exist and be writable prior to deployment.

### `iwwpe_remote_receive.cfg`

Customize the `iwwpe_remote_receive.cfg` file as described below:

- `port` — The port number where OpenDeploy server listens for connections. This must match the `remote_port` value in the `iwwpe_local_send.cfg` file. Customize this value if necessary.
- `teamsite_server` — The `od_client`. Customize this value.
- `allowed_directory` — The IBM WebSphere Application Server installation root. For example, `C:\WebSphere\AppServer`. Customize this value if necessary.

**Note:** The `iwwpe_remote_receive.cfg` and `iwwpe_local_send.cfg` files are OpenDeploy configuration files specific to this solution. If other OpenDeploy configuration files already exist on development or production systems, the sample configuration files should be integrated into the existing files.

### `iwwpe_xml_db.cfg`

Customize the `iwwpe_xml_db.cfg` file as described below:

- In all XML-formatted-data tags, customize the `file` value, if necessary, to reflect the path to XML files (leaving XML file names intact). This path must match with the `remote_directory` value of the `deploy_db_xml` deployment section in the `iwwpe_local_send.cfg` file.



- In all database tags, customize the `db`, `user`, and `password` values to reflect your staging or production database environment.

For DB2 use:

`db` — `//host:port/database_alias`, where `port` is the port number that was used to start the JDBC connection daemon (`db2jstrt`) on the database server host.

`user` — The database login name for the user who has access rights to database tables.

`password` — The user's password.

## The pznload Utility

You may customize the following files for the `pznload` utility:

- `pznResourcesToLoad.txt`
- `pznRulesToLoad.txt`

### **pznResourcesToLoad.txt**

The `pznResourcesToLoad.txt` file contains a list of resources the `pznload` utility will load to the target server.

The file format is one resource path per line. The path should be relative to the personalization resources root folder, `WAS_server_root/personalization/publishedResources`.

By default, the `pznload` utility expects the resources file to be named `pznResourcesToLoad.txt` and to be located in the same directory as the `pznload` utility. To reject the default and specify another resources file name and location, use the `-reslistfile` option in the `pznload` utility.

### **pznRulesToLoad.txt**

The `pznRulesToLoad.txt` file contains a list of rules the `pznload` utility will load to the target server.

The file format is one rule path per line. The path should be relative to the personalization rules root folder, `WAS_server_root/personalization/publishedRules`.

By default, the rules file is expected to be named `pznRulesToLoad.txt` and to be located in the same directory as the `pznload` utility. To reject the default and specify another rules file name and location, use the `-rulelistfile` option in the `pznload` utility.

### **pznload Utility Usage**

The `pznload` utility is used to register the resources and load the rules into the personalization engine. Utility usage and syntax are provided below.

```
pznload <servername> <appserver_root> {-verbose} {-logfile fullpathfilename.xxx}  
{-rulelistfile fullpathfilename.xxx} {-reslistfile fullpathfilename.xxx}
```

|  |   |
|--|---|
| <code>&lt;servername&gt;</code>                    | production server name  |
| <code>&lt;appserver_root&gt;</code>                | application server root path (e.g.<br>C:\WebSphere\AppServer) |
| <code>-verbose(*)</code>                           | enable verbose output   |
| <code>-logfile filename(*)</code>                  | enable logging  |
| <code>-rulelistfile fullpathfilename.xxx(*)</code> | specify rules list file                                       |
| <code>-reslistfile fullpathfilename.xxx(*)</code>  | specify resources list file                                   |

(\*) optional

Consult the `Readme.txt` file in the `WAS_server_root/personalization/publishToProduction` directory for more information about the `pznload` utility.

## **iwwpe\_backup\_data.sql**

The `iwwpe_dump_load.pl` file uses the SQL script `iwwpe_backup_data.sql` to back up the production data and clear the tables before loading new data from XML dump files through DataDeploy. To back up tables other than `articles` and `companynews`, add a section to this script for each table you wish to back up.

Customize the following values:

- `prod-db_alias` — The production database alias.
- `user db2admin` — The account that has rights to drop and create tables and perform select, insert, and delete transactions on the production schema.
- `using db2admin` — The password for the above account.

## **iwwpe\_restore\_data.sql**

If DataDeploy fails to deploy all content to the production database, you can restore the data manually using the `iwwpe_restore_data.sql` script. This script will restore the original data from backup tables created by `iwwpe_backup_data.sql`. To restore tables other than `articles` and `companynews`, add a section to this script for each table you wish to restore.

Customize the following values:

- `prod-db_alias` — The production database alias.
- `user db2admin` — The account that has rights to perform select, insert, and delete transactions on the production schema.
- `using db2admin` — The password for the above account.

## Client Components Configuration

Follow the directions in this section to customize the OpenDeploy and DataDeploy configuration files listed below:

- `iwwpe_local_send.cfg`
- `iwwpe_db_xml.cfg`

**Note:** All directories must exist and be writable prior to deployment.

### `iwwpe_local_send.cfg`

The `iwwpe_local_send.cfg` file consists of several deployment sections, each discussed individually below. Customize them according to the directions that follow.

Customize the first section of the file as follows:

- `hostname` — The `od_client`. Customize this value.
- `remote_server` — The `od_server`. Customize this value.
- `remote_port` — The port number where OpenDeploy server listens for connections. Must match with `port` value in the `iwwpe_remote_receive.cfg` file. Customize this value if necessary.

Except as otherwise noted, all deployment sections have the following parameters in common:

- `area` — The general source directory containing files for the particular deployment. Customize this value, if necessary.
- `local_directory` — The exact source directory.
- `remote_directory` — The exact destination directory. Customize this value, if necessary, according to specific instructions given in each deployment section below.

### `deployment=deploy_jsp`

This section deploys JSP pages and static files. Customize the following values, if necessary:

- `remote_directory` — Your production webapp docroot.

- `source_exclude` — Exclude the directory under `local_directory` from this deployment.

#### **deployment=deploy\_classes**

This section deploys servlets and other class files. Customize the following value, if necessary:

`remote_directory` = your production webapp servlets classpath

**Note:** OpenDeploy only transfers the servlet class files to production. You must register new servlets (if required) and restart necessary Web applications on the production host.

#### **deployment=deploy\_db\_xml**

This section deploys database content (`articles` and `companynews` tables) from the development database to the production database. It includes four Deploy and Run (DNR) calls. Customize the following values, if necessary:

- `area` = general source directory, where to find the XML files for this deployment. Note that this is also the first parameter of the `iwwpe_dump_load.pl` DNR script. If you change this value, you must change the DNR script call as well.
- `local_directory` = exact source directory. Note that this is also the second parameter of the `iwwpe_dump_load.pl` DNR script. If you change this value, you must change the DNR script call as well.
- `remote_directory` = exact destination directory. The value must match with the `file` path in XML-formatted-data tags in the `iwwpe_xml_db.cfg` file.

The four DNR calls in this deployment section are reproduced below as they appear in code, followed by brief descriptions and customization instructions.

```
deploy_run_script="perl.exe <iw-home>\opendeploy\bin\iwwpe_dump_load.pl
C:\database_dump dump_dir dump"
    as=administrator
    when=client_before_deploy
;
```

Before copying files, OpenDeploy runs the `iwwpe_dump_load.pl` script as the administrator to dump the content of the `articles` and `companynews` tables into XML files on `od_client`. Files will be created in the `C:\database_dump\dump_dir` directory. To customize this call, see the descriptions of `area` and `local_directory` above.

```
deploy_run_script="perl.exe <iw-home>\opendeploy\bin\iwwpe_dump_load.pl
C:\database_dump dump_dir clean"
    as=administrator
    when=client_after_deploy
;
```

After copying files, OpenDeploy runs the `iwwpe_dump_load.pl` script as the administrator to clean the dump directory (`C:\database_dump\dump_dir`) on `od_client`. To customize this call, see the descriptions of `area` and `local_directory` above.

```
deploy_run_script="perl.exe <od-home-on-
prodserver>\bin\iwwpe_dump_load.pl <was-root-on-
prodserver>\hosts\default_host\<production-webapp-docroot>\db_xml
load_dir clean"
    as=administrator
    when=server_before_deploy
;
```

Before copying files, OpenDeploy runs the `iwwpe_dump_load.pl` script as the administrator to clean the load directory (for example,

`C:\WebSphere\AppServer\hosts\default_host\Prod\db_xml\load_dir`) on `od_server`. To customize this call, see the description of `remote_directory`.

```
deploy_run_script="perl.exe <od-home-on-
prodserver>\bin\iwwpe_dump_load.pl <was-root-on-
prodserver>\hosts\default_host\<production-webapp-docroot>\db_xml
load_dir load"
    as=administrator
    when=server_after_deploy
;
;
```

After copying files, OpenDeploy runs the `iwwpe_dump_load.pl` script as the administrator to load the XML files into the `articles` and `companynews` tables on `od_server`. XML files are located in, for example,

`C:\WebSphere\AppServer\hosts\default_host\Prod\db_xml\load_dir`. To customize this call, see the description of `remote_directory`.

For `iwwpe_dump_load.pl` usage, see “`iwwpe_dump_load.pl` Usage” on page 86.

### **deployment=deploy\_rules**

This section deploys and registers personalization rules. If necessary, customize the values in this section according to the common parameter values given above. The DNR call in this deployment section is reproduced below as it appears in code, followed by a brief description.

```
deploy_run_script=cmd /C "cd <was-root-on-  
prodserver>\personalization\publishToProduction && pznload.bat prodserver  
<was-root-on-prodserver> -logfile <was-root-on-  
prodserver>\personalization\publishToProduction\pznloadrules.log -  
reslistfile <was-root-on-  
prodserver>\personalization\publishToProduction\dummyres.txt "  
    as=administrator  
    when=server_after_deploy  
;  
;
```

After copying files, this script runs the `pznload` utility as the administrator to register rules on `od_server`. Consult the `Readme.txt` file in the `install/websphere/pznload` directory for more information about the `pznload` utility.

## deployment=deploy\_resources

This section deploys and registers personalization resources. If necessary, customize the values in this section according to the common parameter values given above. The DNR call in this deployment section is reproduced below as it appears in code, followed by a brief description.

```
deploy_run_script=cmd /C "cd <was-root-on-  
prodserver>\personalization\publishToProduction && pznload.bat prodserver  
<was-root-on-prodserver> -logfile <was-root-on-  
prodserver>\personalization\publishToProduction\pznloadresources.log -  
rulelistfile <was-root-on-  
prodserver>\personalization\publishToProduction\dummyrule.txt "  
    as=administrator  
    when=server_after_deploy  
    ;  
    ;
```

After copying files, this script runs the `pznload` utility as the administrator to register resources on `od_server`. Consult the `Readme.txt` file in the `install/websphere/pznload` directory for more information about the `pznload` utility.

## iwwpe\_dump\_load.pl Usage

Usage: `iwwpe_dump_load.pl <area_top> <dump_dir> <action>`

`area_top`            the absolute path to top of area directory

`dump_dir`           the relative path to dump/load directory in area

NOTE: Please use UNIX-style path notation for all paths.

`action`            'dump' do DataDeploy db-to-xml  
                  'load' do DataDeploy xml-to-db  
                  'clean' to clean working dir containing xml files



## iwwpe\_db\_xml.cfg

Customize the `iwwpe_db_xml.cfg` file as described below:

- In all XML-formatted-data tags, customize the `file` value, if necessary, to reflect the path to XML files (leaving XML file names intact). This path must match the `area-top` parameter in all `iwwpe_dump_load.pl` calls for the client in the `iwwpe_local_send.cfg` file. This directory must exist prior to deployment and it must be writable.
- In all database tags, customize `db`, `user`, and `password` values to reflect your development database environment.

For DB2 use:

`db = //host:port/database_alias`, where `port` is the port number that was used to start the JDBC connection daemon (`db2jstrt`) on the database server host.

`user = database login name for user who has access rights to database tables.`

`password = user's password.`

## OpenDeploy Invocation

On `od_server`, Windows platform:

```
SET PATH=<od_server_root>\bin;<od_server_root>\iw-perl\bin\MSWin32-  
x86;%PATH%  
cd <od_server_root>\conf  
iwdeploy -S -fd iwwpe_remote_receive.cfg
```

On `od_server`, UNIX platform:

```
export PATH=<od_server_root>/bin:<path_to_perl_binary>:$PATH  
export LD_LIBRARY_PATH=<od_server_root>/lib/perl5/5.00503/sun4-solaris/  
CORE  
cd <od_server_root>/conf  
iwdeploy -S -fd iwwpe_remote_receive.cfg
```

On *od\_client*, Windows platform:

```
SET PATH=<od_client_root>\bin;<od_client_root>\iw-perl\bin\MSWin32-  
x86;%PATH%  
cd <od_client_root>\conf  
iwdeploy -fs iwwepe_local_send.cfg deploy_jsp  
iwdeploy -fs iwwepe_local_send.cfg deploy_classes  
iwdeploy -fs iwwepe_local_send.cfg deploy_db_xml  
iwdeploy -fs iwwepe_local_send.cfg deploy_rules  
iwdeploy -fs iwwepe_local_send.cfg deploy_resources
```

On *od\_client*, UNIX platform:

```
export PATH=<od_client_root>/bin:<path_to_perl_binary>:$PATH  
export LD_LIBRARY_PATH=<od_server_root>/lib/perl5/5.00503/sun4-solaris/  
CORE  
cd <od_client_root>/conf  
iwdeploy -fs iwwepe_local_send.cfg deploy_jsp  
iwdeploy -fs iwwepe_local_send.cfg deploy_classes  
iwdeploy -fs iwwepe_local_send.cfg deploy_db_xml  
iwdeploy -fs iwwepe_local_send.cfg deploy_rules  
iwdeploy -fs iwwepe_local_send.cfg deploy_resources
```

## How It Works

The OpenDeploy client controls deployment. Each deployment section in the *iwwepe\_local\_send.cfg* file deploys a particular component of the personalization application. You can create a batch file containing all deployments or invoke each deployment separately.

- `deploy_jsp` transfers dynamic (.jsp) and static (.html, .gif, and so on) application pages. The destination directory for these files is usually the webapp doc root.
- `deploy_classes` transfers application servlets (.class files). The destination directory for these files can be any element of the classpath of the webapp.
- `deploy_db_xml` deploys database-based assets of the personalized application. The DNR feature invokes the *iwwepe\_dump\_load.pl* script before and after deployment on the *od\_client* and the *od\_server* to invoke DataDeploy or to perform a cleanup.

The `iwwpe_dump_load.pl` script is a wrapper that invokes DataDeploy to extract database content into XML dump files, to load XML dump files into the database, or to clean the dump or load area. The configuration files for the DataDeploy invocation have been prepped to dump or load data from or into the `articles` and `companynews` tables. OpenDeploy transfers XML dump files from the dump directory into the load directory.

`iwwpe_dump_load.pl` calls the `iwwpe_backup_data.sql` script before it invokes DataDeploy. This script connects to the production database and creates the backup tables `articles_save` and `companynews_save` for the `articles` and `companynews` tables, respectively. The script then inserts all records from the original tables to backup tables and clears the original tables. If DataDeploy fails, you can restore the original data manually using the `iwwpe_restore_data.sql` script.

- `deploy_rules` deploys personalization rules. After deployment DNR invokes the `pznload` utility, which loads the rules into the personalization engine through the `PznRulePublisher` servlet.
- `deploy_resources` deploys personalization resources. After deployment DNR invokes the `pznload` utility, which registers resources with the personalization engine through the `ImportServlet` servlet.

If deployment fails, the entire deployment should be restored to the previous state.



INTERWOVEN